University of Alberta

Analysis and Extension of COCOMO Software Cost Estimation Model

by

Nan Sun

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta
Spring 2003

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-82351-2

Canada

University of Alberta

Library Release Form

Name of Author: Nan Sun

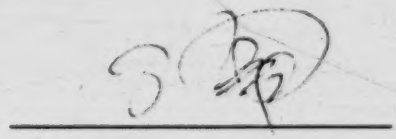
Title of Thesis: Analysis and Extension of COCOMO Software Cost Estimation Model

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Nan Sun
9003—112 Street, Edmonton
Alberta, Canada
T6G 2C5

Date: Jan 6, 2003

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled Analysis and Extension of COCOMO Software Cost Estimation Model submitted by Nan Sun in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor: Dr. Petr Musilek

Dr. Marek Reformat

Dr. Ioanis Nikolaidis

Date: Jan. 6, 2003

Abstract

In this thesis, selected cost estimation models are considered and their sensitivity to imprecision of their parameter estimates is studied. The results of sensitivity analysis are then used as a starting point to enhance these models to acknowledge and directly manipulate such imprecision. The methods employed here stem from the technology of fuzzy systems, allowing direct expression of the imprecision and manipulating it in a way that provides for enhanced information about the imprecision/usefulness of the resulting estimates.

The first model, developed using fuzzy regression, is interesting but not practical due to very large spread of the estimates. The second model, developed using fuzzy arithmetic, provides cost estimates in the form of fuzzy numbers carrying both the modal value of the estimated effort and a measure of its precision. In this case, the spread of the estimates is reasonable and provides valuable information regarding reliability of the estimates.

Acknowledgements

I would first like to thank Dr. Petr Musilek, under whose supervision this work was carried out, for his assistance, guidance, inspiration, discussions, and encouragement throughout the development of this work.

Thanks also go to people I work with at University of Alberta for their support and assistance in the preparation of this thesis—Illiyan Kaytazov, Snezana Djokic, Gabriel Jarillo, Alexander Medin and Guillermo Barreiro.

Finally, I would like to thank my parents, Xiao-Feng and Xiang-Yun, and my sister Pearl. Their continued encouragement and support has made the task of completing this thesis possible.

Table of contents

1. INTRODUCTION 1

1.1. GOALS OF THIS THESIS 1

1.2. STRUCTURE OF THIS THESIS 2

2. A REVIEW OF SOFTWARE COST ESTIMATION MODELS 4

2.1. PRICE MODEL 4

2.2. COCOMO 5

2.3. FUNCTION POINTS ANALYSIS 6

2.4. SLIM 8

2.5. SOFTCOST-R & SOFTCOST-ADA 11

2.6. BEFORE YOU LEAP (BYL) 14

2.7. ESTIMACS 15

2.8. SPQR-20 15

2.9. BIS-ESTIMATOR 16

2.10. SUMMARY 16

3. COCOMO 18

3.1. BASIC 18

3.2. INTERMEDIATE 19

3.3. ADVANCED 22

3.4. COCOMO II 23

3.4.1 The Application Composition Model 24

3.4.2 The Early Design Model 26

3.4.3 The Post-Architecture Model 28

3.5. SUMMARY 30

4. SENSITIVITY OF COCOMO II MODEL 31

4.1. SOFTWARE COST ESTIMATION 31

4.2. COCOMOII MODEL 32

4.3. STUDY OF THE SENSITIVITY 33

4.3.1 Mathematical analysis *33*
4.3.2 Monte Carlo Simulation *36*
4.3.3 Error Propagation *39*
4.4. SUMMARY 48
5. FUZZY REGRESSION OF COCOMO MODEL 49
5.1. FUZZY LINEAR REGRESSION 50
5.1.1 Least Square Linear regression and robust regression *51*
5.1.2 Fuzzified model *53*
5.1.3 Linear programming *53*
5.2. POLYNOMIAL FUZZY REGRESSION 57
5.3. SUMMARY 60
6. FUZZY EXTENSION OF COCOMO MODEL 61
6.1. SOURCES OF UNCERTAINTY IN THE COST ESTIMATION MODELS 61
6.2. FUZZY NUMBERS 62
6.2.1 Triangular Fuzzy Numbers *62*
6.2.2 Parabolic Fuzzy Numbers *63*
6.2.3 Fuzziness of Fuzzy Numbers *64*
6.2.4 Fuzzy Arithmetic *64*
6.3. SIMPLE F-COCOMO MODEL 65
6.3.1 Simple f-COCOMO Model: Triangular Numbers *66*
6.3.2 Simple f-COCOMO Model: Parabolic Numbers *67*
6.4. AUGMENTED F-COCOMO MODEL 71
6.5. SUMMARY 73
7. CONCLUSION 75
8. REFERENCES 78

List of Tables

TABLE 1: FUNCTION POINT COMPLEXITY WEIGHTS 7

TABLE 2: COMPONENTS OF THE TECHNICAL COMPLEXITY FACTOR 7

TABLE 3: EFFORT FOR THREE MODES OF BASIC COCOMO 19

TABLE 4: EFFORT PARAMETERS FOR THREE MODES OF INTERMEDIATE COCOMO 19

TABLE 5: SOFTWARE DEVELOPMENT EFFORT MULTIPLIERS 20

TABLE 6: ANALYST CAPABILITY EFFORT MULTIPLIER FOR DETAILED COCOMO 23

TABLE 7: OBJECT POINT COMPLEXITY LEVELS FOR SCREENS 24

TABLE 8: OBJECT POINT COMPLEXITY LEVELS FOR REPORTS 25

TABLE 9: COMPLEXITY WEIGHTS FOR OBJECT POINTS 25

TABLE 10: AVERAGE PRODUCTIVITY RATES BASED ON DEVELOPER'S EXPERIENCE AND THE ICASE MATURITY/CAPABILITY 26

TABLE 11: PROGRAMMING LANGUAGE LEVELS AND RANGES OF SOURCE CODE STATEMENTS PER FUNCTION POINT 27

TABLE 12: EARLY DESIGN COST DRIVERS 27

TABLE 13: COCOMOII SCALE FACTORS 28

TABLE 14: POST-ARCHITECTURE COST DRIVERS 29

TABLE 15: SENSITIVITY OF EFFORT TO EFFORT MULTIPLIERS 34

TABLE 16: SENSITIVITY OF EFFORT TO SIZE 35

TABLE 17: SENSITIVITY OF EFFORT TO EXPONENT SCALE FACTORS 35

TABLE 18: AVERAGE SENSITIVITY TO THREE GROUPS OF PARAMETERS 36

TABLE 19: TOLERABLE INPUT ERRORS FOUND USING MONTE CARLO SIMULATION 38

TABLE 20: STANDARD DEVIATIONS OF PM FOR INCREASING LEVELS OF UNCERTAINTY L (PROJ. 1) 45

TABLE 21: TOLERABLE INPUT ERROR (TIE) OBTAINED BY ERROR PROPAGATION 46

TABLE 22: TOLERABLE INPUT ERROR OBTAINED BY ERROR PROPAGATION 47

TABLE 23: AVERAGE VALUE OF POSSIBILITY MEASURES OBTAINED USING SIMPLE F-COCOMO MODEL FOR VARIOUS CATEGORIES OF SOFTWARE (CAT) AND APPLICATION DOMAINS (APP). THE DATA IN PARENTHESES INDICATE THE NUMBER OF CASES FOR EACH GROUP 70

List of figures

FIGURE 1: A REYLEIGH CURVE 9

FIGURE 2: MONTE CARLO SIMULATION FOR SINGLE INPUT ERROR IN: (A) SIZE, (B) EFFORT, (C),EXPONENT SCALE FACTORS 37

FIGURE 3: ERROR PROPAGATION 40

FIGURE 4: LINEAR REGRESSION AND ROBUST REGRESSION 52

FIGURE 5: LINEAR AND ROBUST REGRESSION-CLOSER LOOK 52

FIGURE 6: FUZZY REGRESSION 54

FIGURE 7: LINEAR REGRESSION 56

FIGURE 8: IMPROVED LINEAR REGRESSION 57

FIGURE 9: POLYNOMIAL FUZZY REGRESSION 58

FIGURE 10: POLYNOMIAL REGRESSION RESULT 59

FIGURE 11: THE SIZE OF A SMALL SOFTWARE PROJECT MODELED AS A FUZZY SET 62

FIGURE 12: TRIANGULAR AND PARABOLIC FUZZY NUMBERS 63

FIGURE 13: SIMPLE F-COCOMO MODEL WITH TRIANGULAR FUZZY NUMBERS 67

FIGURE 14: SIMPLE F-COCOMO MODEL WITH TFN FOR DIFFERENT DEVELOPMENT MODES 69

FIGURE 15: HISTOGRAMS OF POSSIBILITIES OF EACT 69

FIGURE 16: SPREAD D VS. COVERAGE OF EACT 70

FIGURE 17: PARAMETERS A AND B AS DISCRETE FUZZY NUMBERS 71

FIGURE 18:AUGMENTED F-COCOMO MODEL WITH TRIANGULAR FUZZY NUMBERS 73

1. Introduction

Software cost estimation, the process of predicting the amount of effort required to build a software system, has been the key issue of the software project management field. Reliable and accurate estimates of the software development cost and effort are needed throughout the software development cycle to determine feasibility of software projects and to provide for appropriate allocation or reallocation of available resources. Preliminary estimates are required to determine the feasibility of a project. Detailed estimates are needed to assist with project planning. The actual effort for individual tasks is compared with estimated and planned values, enabling project managers to reallocate resources when necessary. It is greatly desired and required that the task of software cost estimation be performed very early in the development cycle.

1.1. Goals of this Thesis

There are models and tools that are used to address and solve all kinds of problems in software cost estimation. However there is a clear need of enhancement of these models in terms of precision, feasibility and convenience of their use. In general, these models are based on measuring certain size or function related attributes of software and relating these measurements to the cost or effort necessary for its development. Nevertheless, the models depend greatly on the amount of the reliable information among which many are subjective evaluations affected by the uncertainty and imprecision. This thesis tries to analyze the estimation model data to find the sensitivity features of the model, and deduce extended models that improve the overall value of the cost estimation process by

letting the model aware of the uncertainty of the input and have it presented directly as fuzzy input so as to observe the corresponding uncertainty of the output intuitively.

1.2. Structure of this thesis

The second section of this thesis provides a review of software cost estimation models and tools that have been developed throughout the past several decades. It can be seen that not all models have been successful in terms of their wide spread use by researchers and industry practitioners. One model that attracts more attention is called COCOMO (Boehm, 1997). This model is well documented and backed by well-developed estimation tools, and has been adopted by many practitioners. Hence this thesis is focused on discussing this model and its variations. The third section evaluates the influence and sensitivity contribution of model parameters to the cost estimates. Since the accuracy of cost estimation depends on how precisely the parameters are evaluated/determined, there is naturally an interest in finding how they impact the estimation both individually and cumulatively. In other words, how sensitive the cost estimation is to the noise/errors inherent in these parameters. Several experimental methods are employed including mathematical analysis and simulation techniques.

In an attempt to capture the uncertainty present in the COCOMO dataset, a fuzzy regression model has been constructed. The details of the model are described in section 5. The results obtained using Fuzzy Regression show excessive amounts of uncertainty in the data, rendering the regression model unusable. Therefore, another fuzzy extension of the COCOMO model is proposed in section 6. It considers the uncertainty as an inherent

part of the estimation process, accepts it, and provides the estimate in the form of possibility distributions instead of a single numerical value. Section 7 brings a summary of the thesis and provides some important conclusions.

2. A review of software cost estimation models

Since 1960's, numerous software cost methodologies and technologies have been developed that are currently used academically and commercially. Many of the tools involved were introduced by Albrecht, Boehm and Putnam (Albrecht, 1979). By looking at the correlations between cost trends and measurable parameters which are indicated by analysis of historical project data, a wide range of models have been developed. These are used to access, predict, and control software costs on a real-time basis. More models are mathematic functions that compute cost from a number of variables.

2.1. PRICE Model

PRICE model (Hamaker, 1994) is considered as a milestone in cost estimation, although it is mainly used for hardware. It was first developed by Frank Freiman in RCA, and then became marketed and commercialized by RCA. The key contribution of Freiman is to see hardware production costs as processes governed by logical interrelationship among a handful of key variables. He derived the algorithms using his intuitive rules inspired by his engineering experience. By running the model backward to find out what settings of variables gave the known cost, the model could then be calibrated to a specific organization's historical track record. Once calibrated, the model could be run forward using a rich set of factors to predict the cost of future projects. PRICE models first were applied to aerospace industry although they are useful for a wide range of other industries. After observing PRICE models' potential in Space Shuttle system cost estimation, NASA further encouraged Freiman to market the models and provided him more data to calibrate the model. The success of the PRICE model inspired the

development of several other commercial cost models with application to hardware, and software life cycles.

Compared to PRICE model, COCOMO and FPA have the disadvantage that the underlying concepts and ideas are not publicly defined and the users are presented with the model as a black box. The user of PRICE sends the input to a time--sharing computer in the USA, UK, or France and gets back his estimates immediately. Despite this disadvantage and the high rental price, there are many users, especially in America.

By the late 1970s and into the mid-1980s, NASA began to realize that project cost was a serious problem. Overruns were worse than ever despite better databases, better models, better estimators, and more stringent Headquarters reviews. Hum Mandell, assisted by Richard Whitlock and Kelley Cyr, initiated analyses of this problem. Making imaginative use of the PRICE model, they found that the complexity of NASA projects had been steadily increasing mostly because of NASA's culture to be exotically expensive rather than practically affordable. At the same time, NASA management realized that estimates of future projects should consider the steadily increasing complexity of NASA projects.

2.2. COCOMO

COCOMO (Constructive Cost Model) is derived by Boehm using data from 63 sets of projects at TRW, a consulting company based in California (Fenton, 1997). COCOMO is a relatively very well documented and straightforward model based on system size related inputs and a number of cost drivers that affect cost estimation. The original

COCOMO model was first published in 1981 (Boehm, 1981) and it is sets of equations used for estimating the influence of 15 cost drivers on the development effort (or cost). Before this can be done, an estimate of the software size must be available. The equations represent the relations between size and effort and between effort and development time. More detailed discussion on COCOMO models will be presented in the next section.

2.3. Function Points Analysis (FPA)

Albrecht, who is from IBM is the developer of FPA, a method by which productivity in software development is measured in an alternative way to the number of lines of code (Albrecht, 1979). This measurement is independent from programming language and fourth generation tools. The idea behind FPA is based on the amount of functionality that the software must fulfil, which is related to the types of data the software is involved. Function point computations are based on the calculation result of an ***unadjusted function point count*** (UFC), which are made for the following categories (Fenton, 1997):

- *External inputs* – those items provided by the user that describe distinct application-oriented data (such as file names and menu selections)
- *External outputs* – those items provided to the user that generate distinct application-oriented data (such as reports and messages, rather than their individual components)
- *External inquiries* – interactive inputs requiring a response
- *External files* – machine-readable interfaces to other systems
- *Internal files* – logical master files in the system

There is a complexity scale rating associated with each count as illustrated in Table 1. For each of these five types of counts, the number of simple, average and complex occurrences that are expected in the software is evaluated. By weighting each number

with an appropriate weight, the unadjusted number of function points is obtained, which is multiplied by its corresponding complexity weight and the results are summed to provide the UFC.

Table 1: Function point complexity weights

	Weighting Factor		
Count	Simple	Average	Complex
External inputs	3	4	6
External outputs	4	5	7
External inquiries	3	4	6
External files	7	10	15
Internal files	5	7	10

The adjusted function point count (FP) is then calculated by multiplying the UFC by a technical complexity factor (TCF), whose components are listed in Table 2.

Table 2: components of the technical complexity factor

F1	Reliable back-up and recovery	F2	Data communications
F3	Distributed functions	F4	Performance
F5	Heavily used configuration	F6	Online data entry
F7	Operational ease	F8	Online update
F9	Complex interface	F10	Complex processing
F11	Reusability	F12	Installation ease
F13	Multiple sites	F14	Facilitate change

Each component value is an integer rated from 0 to 5, indicating how critical the component affect the system. 0 means the component has no influence on the system and 5 means the component is essential (Pressman, 1997). The TCF can then be calculated as:

$$TCF = 0.65 + 0.01(\sum_{i=1}^{14} F_i) \quad (1)$$

The resulting TCF varies from 0.65 (if each *Fi* is set to 0) to 1.35 (if each *Fi* is set to 5) (Fenton, 1997). The final function point calculation is:

$$FP = UFC \times TCF \quad (2)$$

2.4. SLIM

Based on the work of Norden, SLIM (Software Life Cycle Management) is developed as a constraint model by Putnam in 1974, and it's applied to projects exceeding 70,000 lines of code (Putnam, 1978). For many projects at IBM, Norden concluded and plotted frequency distributions, which indicated the amount of people allocated to the development and maintenance of a software product respectively during the whole project development life-cycle. Excitingly, the curves he made by empiricism resemble the Rayleigh curves very much. Unfortunately, he failed to explain the shape of the effort curve. Based on the assumptions of Norden, Putnam assumes that effort for software projects is a Rayleigh curve distribution, implying that staffing rises smoothly during the project and then drops sharply during acceptance testing. The SLIM model is expressed as two equations describing relation between the development effort and the schedule. The first equation, called the *software equation*, states that development effort is proportional to the cube of the size and inversely proportional to the fourth power of the

development time. The second equation, the *manpower-buildup* equation, states that the effort is proportional to the cube of the development time.

Norden observed that the Rayleigh distribution is a good approximation of the manpower curve for various hardware development processes, and hence he found Norden-Rayleigh curve describing manpower as a function of time (Pillai, 1997).

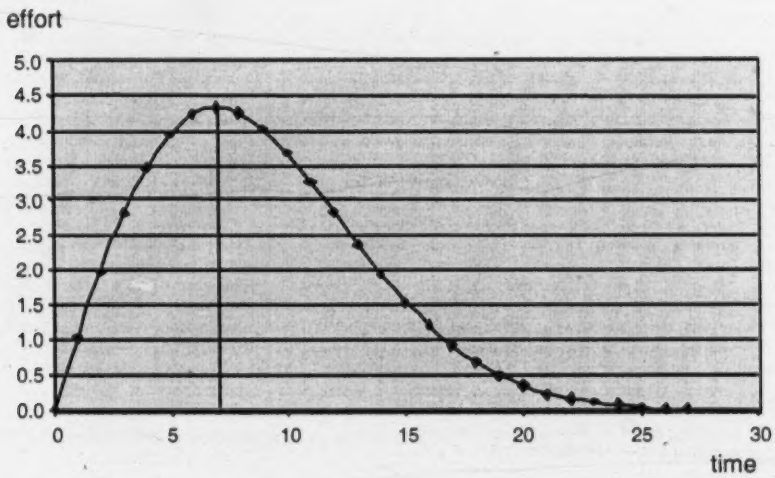


Figure 1: A Reyleigh curve

SLIM utilizes separate Rayleigh curves for design and code, test and validation, maintenance, and management. A Rayleigh curve is shown in Figure 1. Development (design and code) effort is assumed to represent only 40 percent of the total life cycle cost. Requirements specification is not included in the model. Estimation using SLIM is not expected to take place until design and coding.

Since Norden's original observations were not based on theory but rather on observations, and the data his finding based on are basically hardware project data, there are some suspicions about the use of a Rayleigh curve as a basis for cost estimation (Pillai, 1997). Software projects are different from hardware projects in that they are staffed differently and sometimes exhibit a rapid manpower buildup which invalidate the SLIM model for the beginning of the project.

Putnam based his Software Equation on the basic Rayleigh curve formula and his empirical observations about productivity (Fenton, 1997). The software equation is expressed as:

$$Size = CE^{\frac{1}{3}} \times t^{\frac{3}{4}} \tag{3}$$

where C is a technology factor, E is the total project effort in person years, and t is the elapsed time to delivery in years.

The technology factor is a composite cost driver involving 14 components. It primarily reflects:

- Overall process maturity and management practices
- The extent to which good software engineering practices are used
- The level of programming languages used
- The state of the software environment
- The skills and experience of the software team
- The complexity of the application

The software equation includes a fourth power and therefore has strong implications for resource allocation on large projects. Relatively small extensions in delivery date can result in substantial reductions in effort (Pressman, 1997).

Putnam's Manpower-Buildup Equation:

$$D = \frac{E}{t^3} \tag{4}$$

is introduced to find out effort estimation, where D is a constant called manpower acceleration, E is the total project effort in years, and t is the elapsed time to delivery in years. The manpower acceleration D is 12.3 for new software with many interfaces and interactions with other systems, 15 for standalone systems, and 27 for reimplementations of existing systems.

Using the software and manpower-buildup equations, effort can be deduced as (Fenton, 1997):

$$E = \left(\frac{S}{C}\right)^{\frac{9}{7}} \times D^{\frac{4}{7}} \tag{5}$$

This equation is interesting because it shows that effort is proportional to size to the power 9/7 or ~1.286, which is similar to Boehm's factor which ranges from 1.05 to 1.20.

2.5. SOFTCOST-R & SOFTCOST-ADA

SoftCost-R is applicable to all types of programs and considers all phases of the software development cycle. The model is available for lease on IBM PC's. It contains a database of over 1500 data processing, scientific and real-time programs. SoftCost-R model was developed by Dr. Don Reifer based on the work of Dr. Robert Tausworthe of the NASA

Jet Propulsion Laboratory (Collins, 1994). SoftCost is now marketed by Resource Calculations, Inc. of Englewood, Colorado.

Based on a database of over 150 completed Ada written projects, SoftCost-Ada has been developed to match the new object-oriented and reuse paradigm which are emerging not only in Ada, but also C++ and other object-oriented techniques.

There are more than 30 inputs for SoftCost-R consisting of the universal cost estimation input size, other four COCOMO-like category inputs, and some unique ones including use of peer reviews, customer experience, and degree of standardization. The sizing model of SoftCost-R is more complex than COCOMO because it involves not only the value in SLOC or computed function points, but also a more sophisticated sizing pattern : besides reused code, sizes of modules added or deleted may also be included. Each input except size requires a rating ranging from "very low" to "extra high", with "nominal" ratings having no effect on effort calculations. SoftCost-R also uses COCOMO inputs to compare the results of SoftCost-R with those of an updated version of COCOMO.

By using powerful multivariable differential calculus to develop solutions relying on the T-W probability distribution, SoftCost-R delivers more sophisticated functionality than COCOMO. It allows users to perform "what-if" analysis and observe what would happen to schedule if effort were constrained. SoftCost-R is one of the few models for which the mathematical algorithms are completely described in the user's manual. The SoftCost-R equation is:

$$PM = P_0 \times A_1 \times A_2 \times SLOC^B \quad (6)$$

where,

- PM = number of person-months,
- P_0 is a constant factor that may be calibrated,
- A_1 is the "Reifer cost factor" which is an exponential product of nine inputs,
- A_2 is a productivity factor computed from 22 inputs,
- B is an exponent which may be calibrated.

To help the user understand the effect of each input on effort and schedule, the user's manual illustrates values assigned to ratings for all model inputs.

The main outputs of SoftCost-R are effort in person-months, project schedule and productivity value. Other more advanced outputs include several "what if" analysis options, a side-by-side comparison with a recent version of COCOMO, a resource allocation summary for any of three development methods (traditional waterfall, incremental development, or Ada object-oriented), and schedule outputs for Gantt and PERT charts. SoftCost-Ada output composition are similar, and can interface with project planning tools in the same way.

SoftCost-R and SoftCost-Ada both have a calibration file containing values for multiple calibration constants and cost drivers, which acna be changed by users to better describe the user's unique environment, and store alternative calibration and WBS files for different jobs.

SoftCost-R also provides a separate life cycle model for support costs consideration. Besides SoftCost-R development inputs, life cycle inputs also include length of support period, annual change traffic, a sustaining engineering factor, and economic factors. In addition to annual and total support costs as outputs, the life cycle model has optional reports for various staffing options, fixed levels of maintenance, and fixed work force levels.

2.6. Before You Leap (BYL)

BYL is a commercial package based on a link-up between FPA and COCOMO (Gordon, 1986). BYL calculates net function points, which is then translated into source lines of code, taking the language used into consideration. This estimate of the size in SLOC is what COCOMO and the COCOMO part of BYL need as inputs, taking into account the influence on effort of the 15 COCOMO cost drivers, calculates the estimates of costs and time-scale.

BYL is a set of cost estimation models that are used to check all the main factors that affect the development and maintenance of a software project, which may include risk judgements, resource limitation and schedule constraints to the detailed estimate breakdown.

BYL claims to guarantee the users that the estimates will reflect what is really happening to the organization and allow managers to examine performance and productivity factors for management activities other than cost estimation. This is because that BYL models

have all been developed from detailed project data and all are designed to accept actual historical performance into its database.

A range of decisions made against each estimate is kept by BYL so that the managers or customers know exactly the criteria upon which the estimate was made. This makes the managers more confident, prevent errors and help improve the accuracy of the estimates.

2.7. Estimacs

Rubin (Rubin, 1983) (Rubin, 1984) (Rubin, 1985) and Computer Associates (Computer Associates CA-Estimacs User Guide Release 5.0, 1986) developed Extimacs, which is available as a software package. The model consists of nine modules: a function point module; a risk module; an effort module (to estimate development and maintenance effort), etc. The most important and extensive module is Effort. A survey is employed to get from users the answers of 25 input questions, which are partly related to the complexity of the user-organization and partly to the complexity and size of the software to be developed.

2.8. SPQR-20

SPQR has been developed by C. Jones (Jones, 1986). It stands for software Productivity, Quality and Reliability. SPQR is applicable for all kinds of software projects in their estimate of duration, costs and effort to develop software as well as an estimate of maintenance costs. SPQR uses FPA to size the volume of a program. The model is based on an extensive database of past projects. There are four versions of model, SPQR 10, 20,

50 and 100 (the numbers stand for the number of questions the model user has to answer and gives an indication of the degree of refinement of the versions).

2.9. BIS-Estimator

According to the documentation (BIS/Estimator User manual, 1987) BIS-Estimator is completely different from other models as a "knowledge-based tool". This cannot be fully confirmed, because the principles of the model are hided for the most part. The model first gives a rough estimate of duration and effort based on a few input questions. Then a more accurate phase estimate is made for each phase, which is executed during the phase or at the beginning of the phase. Based on the estimates by phase, by means of extrapolation, an estimate of the complete project is made. The model provides a comparison with user-selected projects and evolutionary approach, which makes the estimation process change during software development.

2.10.Summary

The quality of the current estimation techniques is frequently shown to be poor by researchers. More than one techniques and frequent estimation activities during the development of an application are required, especially at the milestones of the development phases. This is because of lack of accurate and reliable estimation techniques combined with the financial, technical, organizational and social risks of software projects. The level of knowledge of the software whose cost we are trying to estimate is growing during a project. A possibility is to use another model during a project, because more information and more accurate information is available.

Successful cost estimation requires collection of data of completed projects. And because the relevant data are different for each organization, it is no solution to use data collected from other organizations. Cost models, estimation by analogy and experts require such consistent data.

Software development should be performed by highly qualified professionals, and such people values individuality in work performance and good professional results. They want to be consulted in decision, work planning and the desired result but not to be disturbed by management during the execution of their work.

Without consultations, it not wise to confront professional developers with a plan and estimate. In consulting the developers not only their expertise is used but also their involvement in the estimation process is increased. This results in a higher commitment than is necessary for the success of a project.

Software cost estimation is much more than a technical problem that can be solved with calculation models, a set of metrics and procedures. The human factors play very important roles. The quality, experience and composition of the project team, the degree in which the project leader can stimulate and commit his developers, affects delivering cost more than the use of rigid calculations.

3. COCOMO

COCOMO consists of three models: *Basic, Intermediate* and *advanced* models. A *Basic* model that is applied early in the project, an *Intermediate* model that is applied after requirements are specified, and an *Advanced* model that is applied after design is complete (Fenton, 1997). All three models share the same basic form:

$$E = a \times S^b \times EAF \tag{7}$$

where E is effort in person months, S is size measured in thousands of lines of code (KLOC), and EAF is an effort adjustment factor having different values for different models (Fenton, 1997). It equals to 1 for Basic model. The factors a and b depend on the development mode. Boehm has defined three development modes:

- *Organic mode* – relatively simple projects in which small teams work to a set of informal requirements.
- *Semi-detached mode* – an intermediate project in which mixed teams must work to a set of rigid and less than rigid requirements.
- *Embedded mode* – a project that must operate within a tight set of constraints.

3.1. Basic

The Basic COCOMO model is a function of program size (Pressman, 1997):

$$E = a \times S^b$$

The factor value a and b of different development modes for the Basic COCOMO model are shown in Table 3 (Boehm, 1981).

Table 3: Effort for three modes of Basic COCOMO

Mode	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

3.2. Intermediate

Besides size, the Intermediate COCOMO model still needs a set of cost drivers as inputs (Pressman, 1997). The Intermediate COCOMO equation is:

$$E = a \times KLOC^{b} \times EAF \tag{8}$$

The values of factors a and b of different development modes for the Intermediate COCOMO model are shown in Table 4 (Boehm, 1981).

Table 4: Effort parameters for three modes of Intermediate COCOMO

Mode	a	b
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

The Effort Adjustment Factor (EAF) is the product of 15 cost drivers which are grouped into four categories: *product*, *computer*, *personnel*, and *project*. Each cost driver is rated on a six-point ordinal scale ranging from low to high importance as shown in Table 5 (Boehm, 1981).

Table 5: Software Development Effort Multipliers

Group	Cost Driver	Description	Rating					
			Very Low	Low	Nominal	High	Very High	Extra High
Product	RELY	Required software reliability	0.75	0.88	1.00	1.15	1.40	-
	DATA	Database size	-	0.94	1.00	1.08	1.16	-
	CPLX	Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
Computer	TIME	Execution time constraint	-	-	1.00	1.11	1.30	1.66
	STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.56
	VIRT	Virtual machine volatility	-	0.87	1.00	1.15	1.30	-
	TURN	Computer turnaround time	-	0.87	1.00	1.07	1.15	-
Personnel	ACAP	Analyst capability	1.46	1.19	1.00	0.86	0.71	-
	AEXP	Applications experience	1.29	1.13	1.00	0.91	0.82	-
	PCAP	Programmer capability	1.42	1.17	1.00	0.86	0.70	-
	VEXP	Virtual machine experience	1.21	1.10	1.00	0.90	-	-
	LEXP	Language experience	1.14	1.07	1.00	0.95	-	-
Project	MODP	Modern programming practices	1.24	1.10	1.00	0.91	0.82	-
	TOOL	Software Tools	1.24	1.10	1.00	0.91	0.83	-
	SCED	Development Schedule	1.23	1.08	1.00	1.04	1.10	-

Product Attributes

RELY (Required Software Reliability): The extent to which the software product must perform its intended functions satisfactorily over a period of time.

DATA (Data Base Size): The degree of the total amount of data to be assembled for the data base.

CPLX (Software Product Complexity): The level of complexity of the product to be developed.

Computer Attributes

TIME (Execution Time Constraint): The degree of the execution constraint imposed upon a software product.

STOR (Main Storage Constraint): The degree of main storage constraint imposed upon a software product.

VIRT (Virtual Machine Volatility): The level of the virtual machine underlying the product to be developed.

TURN (Computer Turnaround Time): The level of computer response time experienced by the project team developing the product.

Personnel Attributes

ACAP (Analyst Capability): The level of capability of the analysts working on a software product

AEXP (Applications Experience): The level of applications experience of the project team developing the software product.

PCAP (Programmer Capability): The level of capability of the programmers working on the software product.

VEXP (Virtual Machine Experience): The level of virtual machine experience of the project team developing the product.

LEXP (Programming Language Experience): The level of programming language experience of the project team developing the product.

Project Attributes

MODP (Use of Modern Programming Practices): The degree to which modern programming practices (MPPs) are used in developing software product.

TOOL (Use of Software Tools): The degree to which software tools are used in developing the software product.

SCED (Schedule Constraint): The level of schedule constraint imposed upon the project team developing the software product.

3.3. Advanced

The Advanced COCOMO model computes effort as a function of program size and a set of cost drivers (the same set as intermediate model) weighted according to each phase of the software lifecycle. The 4 phases used in the detailed COCOMO model are: requirements planning and product design (RPD), detailed design (DD), code and unit test (CUT), and integration and test (IT). The Advanced model applies the Intermediate model at the component level, and then a phase-based approach is used to consolidate the estimate (Fenton, 1997). Each cost driver is broken down by phase as in the example shown in Table 6 (Boehm, 1981).

Table 6: Analyst capability effort multiplier for Detailed COCOMO

Cost Driver	Rating	RPD	DD	CUT	IT
ACAP	Very Low	1.80	1.35	1.35	1.50
	Low	0.85	0.85	0.85	1.20
	Nominal	1.00	1.00	1.00	1.00
	High	0.75	0.90	0.90	0.85
	Very High	0.55	0.75	0.75	0.70

Using the detailed cost drivers (in the same way as the intermediate model by multiplying all the cost driver value), an estimate is determined for each phase of the lifecycle.

3.4. COCOMO II

Although COCOMO befit build-to-specification software projects quite well, COCOMO II can be applicable for a much wider collection of techniques and technologies (Boehm, 1997). The cost estimation model COCOMO II (Boehm, 1999) has been developed as a major revision of the original COCOMO (Boehm, 1981) model by group of industrial and academic affiliates at the University of Southern California. This group reviewed the revised set of COCOMO equations and has been collecting data to evaluate hypotheses to select the significant parameters for COCOMOII. COCOMO II provides up-to-date support for business software, object-oriented software, software created via spiral or evolutionary development models, and software developed using commercial-off-the-shelf application composition utilities (Boehm *et al*, 1997).

The new model specifically deals with the availability of additional information in later stages of a project, the nonlinear costs of reusing software components, and the effects of several factors on the diseconomies of scale (such as the turnover rate of the staff, the team distribution, and the maturity of the development process). It also revises some coefficient values and eliminates interruptions present in the old model.

COCOMOII consists of three submodels, each one offering various fidelities. Listed in increasing fidelity, these submodels are called Applications Composition, Early Design, and Post-architecture models. Until recently, only the last and most detailed submodel, Post-architecture, had been implemented in a calibrated software tool. Hence unless otherwise explicitly indicated, all further references in this thesis to COCOMO II mean the Post-architecture model (Madachy, 1999).

3.4.1 The Application Composition Model

The Application Composition model handles potential high-risk issues such as user interfaces, software/system interaction, performance, and technology maturity. *Object points* are used for sizing instead of the traditional LOC metric.

An initial size measure is determined by counting the number of screens, reports, and third-generation components that will be involved in the application. Each object is rated as simple, medium, or difficult using Tables 7 and 8 (Fenton, 1997).

Table 7: Object point complexity levels for screens

	Number and source of data tables		
Number of views contained	Total <4	Total <8	Total 8+
<3	Simple	Simple	Medium
3-7	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult

Table 8: Object point complexity levels for reports

	Number and source of data tables		
Number of views contained	Total <4	Total <8	Total 8+
<3	Simple	Simple	Medium
3-7	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult

The value in each cell is then weighted according to Table 9. The weights represent the relative effort required to implement an instance of that complexity level (Fenton, 1997).

Table 9: Complexity weights for object points

Object type	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	-	-	10

The weighted instances are summed up to get an object point value. Next reuse is put to consideration. If *r*% of the objects will be reused from previous projects, the number of new object points (NOP) is:

$$NOP = (OBJECT\_POINTS) \times \frac{(100 - r)}{100} \quad (9)$$

A productivity rate (PROD) is determined using Table 10 (Boehm *et al*, 1997).

Table 10: Average productivity rates based on developer's experience and the ICASE maturity/capability

Developers' experience and capability	Very Low	Low	Nominal	High	Very High
ICASE maturity and capability	Very Low	Low	Nominal	High	Very High
PROD	4	7	13	25	50

Effort can then be estimated using the following equation (Boehm *et al*, 1997):

$$E = \frac{NOP}{PROD} \quad (10)$$

3.4.2 The Early Design Model

The Early Design model evaluates alternative software and system architectures and concepts of operation. An unadjusted function point count (UFC) is used for sizing. This value is converted to LOC according to Table 11 (Jones, 1996).

Table 11: Programming language levels and ranges of source code statements per function point

Language	Level	Min	Medium	Max
Machine language	0.10	-	640	-
Assembly	1.00	237	320	416
C	2.50	60	128	170
RPGII	5.50	40	58	85
C++	6.00	40	55	140
Visual C++	9.50	-	34	-
PowerBuilder	20.00	-	16	-
Excel	57.00	-	5.5	-

The Early Design model equation is:

$$E = a \times KLOC \times EAF \tag{11}$$

where a is a constant set to 2.45.

The effort adjustment factor (EAF) is calculated using the 7 cost drivers shown in Table 12 (Boehm *et al*, 1997), in a similar way with the original COCOMO model.

Table 12: Early design cost drivers

Cost Driver	Description	Counterpart Combined Post-Architecture Cost Driver
RCPX	Product reliability and complexity	RELY, DATA, CPLX, DOCU
RUSE	Required reuse	RUSE
PDIF	Platform difficulty	TIME, STOR, PVOL

PERS	Personnel capability	ACAP, PCAP, PCON
PREX	Personnel experience	AEXP, PEXP, LTEX
FCIL	Facilities	TOOL, SITE
SCED	Schedule	SCED

3.4.3 The Post-Architecture Model

The Post-Architecture model is used in the development and maintenance phases. Size is measured with function points or LOC, with reuse and software breakage modifiers. The model uses 17 cost drivers and 5 factors determining the projects scaling component. The 5 factors replace the functionality of development modes (organic, semidetached, embedded) of the original COCOMO model.

The Post-Architecture model equation is:

$$E = a \times KLOC^{b} \times EAF \tag{12}$$

where a is set to 2.94 and b is calculated as:

$$b = 0.91 + 0.01 \times \sum W_i \tag{13}$$

where W is the set of 5 scale factors shown in Table 13 (Boehm *et al*, 1997).

Table 13: COCOMOII scale factors

W_i	Very Low	Low	Nominal	High	Very High	Extra High
Precedentedness	4.05	3.24	2.42	1.62	0.81	0.00
Development/Flexibility	6.07	4.86	3.64	2.43	1.21	0.00
Architecture/Risk Resolution	4.22	3.38	2.53	1.69	0.84	0.00
Team Cohesion	4.94	3.95	2.97	1.98	0.99	0.00
Process Maturity	4.54	3.64	2.73	1.82	0.91	0.00

The EAF is calculated using the 17 cost drivers shown in Table 14 (Boehm *et al*, 1997).

Table 14: Post-Architecture cost drivers

Group	Cost Driver	Description	Rating					
			Very Low	Low	Nominal	High	Very High	Extra High
Product	RELY	Required software reliability	0.75	0.88	1.00	1.15	1.39	-
	DATA	Database size	-	0.93	1.00	1.09	1.19	-
	CPLX	Product complexity	0.70	0.88	1.00	1.15	1.30	1.66
	RUSE	Required reusability		0.91	1.00	1.14	1.29	1.49
	DOCU	Documentation		0.95	1.00	1.06	1.13	
Platform	TIME	Execution time constraint	-	-	1.00	1.11	1.31	1.67
	STOR	Main storage constraint	-	-	1.00	1.06	1.21	1.57
	PVOL	Platform volatility	-	0.87	1.00	1.15	1.30	-
Personnel	ACAP	Analyst capability	1.50	1.22	1.00	0.83	0.67	-
	PCAP	Programmer capability	1.37	1.16	1.00	0.87	0.74	-
	PCON	Personnel continuity	1.24	1.10	1.00	0.92	0.84	-
	AEXP	Applications experience	1.22	1.10	1.00	0.89	0.81	-
	PEXP	Platform experience	1.25	1.12	1.00	0.88	0.81	-
	LTEX	Language and tool experience	1.22	1.10	1.00	0.91	0.84	
Project	TOOL	Software Tools	1.24	1.12	1.00	0.86	0.72	-
	SITE	Multisite development	1.25	1.10	1.00	0.92	0.84	0.78
	SCED	Development Schedule	1.29	1.10	1.00	1.00	1.00	-

3.5. Summary

Software industry is becoming more and more dependent on various cost or effort estimation techniques, these techniques are crucial in early stages of software development projects as they address such important issues as affordability and resource allocation. Among existing algorithmic models of software development effort, COCOMO family of models has found a positive response among industry and practitioners. For this reason, this model has been described here in great detail, including its two variants COCOMO and COCOMOII.

In the following section, these models are subject to throughout examination with goal to determine their sensitivity to imprecision in their independent variables. Subsequently, two possible extensions of COCOMNO models are proposed and critically examined that have potential to improve quality of effort estimation wit these models by accepting and explicitly handling such imprecision.

4. Sensitivity of COCOMO II Model

4.1. Software Cost Estimation

The cost of a software project depends on the nature and characteristics of the project and, therefore, the accuracy of the estimates depends on the amount of reliable information about the product to be developed. Obviously the amount and quality of information available at the very beginning of any project is limited. However, most cost estimation models rely heavily on such data obtained typically by subjective expert evaluations affected by high degree of imprecision and uncertainty. This is also the case for the newest addition to the family of cost estimation models, the recently revised COCOMO II. For this reason, this model has been chosen as a subject of the analytical and experimental treatment described here. The model is briefly described in section 4.2 along with a concise introduction of the software cost estimation models in general. To assess the effect of imprecise evaluations, a comprehensive sensitivity analysis has been performed on the selected model. Results of the sensitivity study are described in great detail in Section 4.3. It describes three complementary approaches to assess the sensitivity of mathematical models. Together, the results obtained using these three techniques provide a complete picture of the sensitivity of the model to the variation of its independent variables. Qualitatively, the results follow our intuition: the most critical independent variable is projected size of the software system. Quantitatively, the results are quite alarming, indicating that sensitivity to the imprecision in size estimates is very high [Musilek, 2002]. A possible way to cope with imprecise information in software cost estimation is indicated in the concluding part of this section.

Software cost estimation is of great interest to software project managers who need to predict project costs during early phases in the software life cycle. As a result, many different models of software cost and effort estimation have been developed and used during the past four decades. They often share a common approach: effort is expressed as a predefined function of one or more variables (such as the size of the product, experience of the development team, level of reuse, etc.).

4.2. COCOMOII Model

COCOMOII model estimates the effort to develop a software system based on its projected size S, effort multipliers EM_i and exponent scale factors W_i

$$PM = 2.94 \prod_{i=1}^{17} EM_i S^{0.91+0.01\sum_{i=1}^{5} W_i} \tag{14}$$

where PM is the effort expressed in person-months, S is the projected size of the software project (expressed in thousands of lines of code KLOC), EM_i (i = 1,2....17) are effort multipliers, and W_i (i = 1,2....5) are exponent scale factors.

There are 17 effort multipliers capturing the characteristics of the software development that affect the effort to complete the project. These weighted multipliers are grouped into four categories (product, platform, personnel and project) and their product is used to adjust the effort. The nominal weight assigned to each multiplier is 1.0. If a rating level has a detrimental effect on effort, its corresponding multiplier is greater than 1.0. Conversely, if the rating level reduces the effort then the corresponding multiplier is less than 1.0.

There are 5 exponent scale factors (precedentedness, development flexibility, architecture/risk resolution, team cohesion and process maturity). They account for the relative economies or diseconomies of scale encountered as a software project increases its size based on different nominal values and rating schemes.

4.3. Study of the Sensitivity

To examine sensitivity of the model (its dependent variable *PM*) to variations of the independent variables (S, EM_i and W_i), three groups of experiments/analytical evaluations have been performed: mathematical analysis, Monte Carlo simulation, and error propagation.

4.3.1 Mathematical analysis

In order to analyze the model sensitivity analytically, the derivatives of the dependent variable with respect to each independent variable are determined to quantitatively evaluate the sensitivities to different input variables. Subsequently, tolerable error values for EM_i, S and W_i are determined later to find out the most critical input variable.

The sensitivity of effort to EM_i can be determined using the following derivative equation

$$\frac{\partial PM}{\partial EM_i} = \prod_{\substack{j=1 \\ j \neq i}}^{17} EM_j \times 2.94 \times S^{0.91+0.01\sum_{j=1}^{5} W_j} . \qquad (15)$$

Based on these equations, the tolerable EM_i relative errors have been determined using available data on 15 projects (Appendix A). The threshold of the relative error of effort has been set to the value 5%. The results are summarized in Table 15.

Table 15: Sensitivity of effort to effort multipliers

P	EM_1	EM_2	EM_3	EM_4	EM_5	EM_6	EM_7	EM_8	EM_9	EM_{10}	EM_{11}	EM_{12}	EM_{13}	EM_{14}	EM_{15}	EM_{16}	EM_{17}
1	5.35%	5.31%	5.35%	5.35%	5.35%	5.35%	5.86%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%
2	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%
3	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%
4	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%
5	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	4.83%	5.00%
6	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%
7	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%
8	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%
9	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%
10	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%
11	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%
12	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%	5.35%
13	4.35%	4.35%	4.01%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.35%	4.33%	4.35%
14	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%
15	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%	5.00%

The tolerable error values range within 4.01% and 5.35% with the average of 4.99%. For the same project data, the values are very close to each other, and the values of a single input across all project data are also quite similar.

The sensitivity of effort to input variable S can be determined by the equation

$$\frac{\partial PM}{\partial S} = \prod_{i=1}^{17} EM_i \times 2.94 \times (0.91 + 0.01 \sum_{i=1}^{5} W_i) \times S^{0.01\sum_{i=1}^{5} W_i - 0.09} \quad (16)$$

and the sensitivity to exponential scale factors W_i by the equation

$$\frac{\partial PM}{\partial W_i} = \prod_{i=1}^{17} EM_i \times 2.94 \times S^{0.91 + 0.01\sum_{\substack{j=1 \\ j \neq i}}^{5} W_j} \times \log(S) \times 0.01 \quad (17)$$

The results, assuming the threshold of the relative error of effort to be again 5%, are summarized in Tables 16 and 17.

Table 16: Sensitivity of effort to size

P.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S [%]	4.45	4.43	4.39	4.85	4.46	4.46	4.46	3.80	4.82	3.75	5.19	5.23	3.78	4.32	4.64

Table 2 shows the tolerable relative errors of S that are quite similar across all 15 projects. They range from 3.75% to 5.23% with the average value of 4.46%. The results for W_i, shown in Table 3, have much wider range (13.99% - 155.80%) and the average value of 40.04%. Four notably large values are encountered for W_1 and W_2 for projects 11 and 12. This means that the preset threshold value of error of effort (5%) is reached only for very large errors (> 100%) of the corresponding coefficients. This could be explained by very small values of these coefficients for given projects. The values of sensitivity to W_4 for projects 11, 12 and 15 are not available (N/A) because the actual values of W_4 in these three projects are zero, thus preventing proper calculations (division by zero).

P.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
W_1	22.01 %	49.71 %	28.20 %	42.14 %	31.72 %	21.43 %	23.31 %	48.40 %	36.40 %	21.51 %	126.9 %	105.4 %	20.93 %	25.05 %	36.54 %
W_2	21.53 %	40.55 %	27.58 %	30.92 %	38.84 %	20.96 %	22.81 %	35.51 %	26.71 %	32.92 %	155.8 %	129.4 %	32.04 %	38.35 %	74.52 %
W_3	15.44 %	19.38 %	32.98 %	55.39 %	37.10 %	25.06 %	27.27 %	31.87 %	47.84 %	23.60 %	55.59 %	46.17 %	30.61 %	27.49 %	80.06 %
W_4	24.92 %	56.29 %	42.51 %	71.57 %	47.82 %	48.52 %	52.79 %	41.11 %	61.83 %	30.44 %	N/A	N/A	39.45 %	47.21 %	N/A
W_5	13.99 %	15.80 %	25.61 %	28.71 %	33.62 %	22.71 %	24.71 %	38.47 %	21.70 %	28.49 %	22.41 %	20.94 %	20.80 %	28.45 %	48.41 %

Table 17: Sensitivity of effort to exponent scale factors

As the results obtained the three types of input variables (EM_i, S and W_i) are quite uniform within each group, they have been grouped and averaged to find the order of impact of their inaccuracy on the error of the estimated effort. For convenience the results are summarized in the following table.

Table 18: Average sensitivity to three groups of parameters

Group Of Parameters	Average Tolerable Error
EM_i	4.99%
S	4.47%
W_i	40.04%

Evidently projected size S is the most critical parameter as the estimates are most sensitive to it. It is closely followed by effort multipliers EM_i whose impact is only slightly smaller. Finally, the exponent scale factors W_i exhibit relative low impact on the accuracy of the estimates. Implications of this finding are explained later in this section taking in account results of other sensitivity analysis methods.

4.3.2 Monte Carlo Simulation

Monte Carlo simulation is applied to a single error input case, and it is assumed that only one input variable has error in one simulation run as shown in Figure 2. The distribution

of the error was initially considered to be normal that is fully justifiable for size S that does not have any strict upper limit of range. Since normal distribution has no limited range, 99.74% of the data fall within the goal range by making the standard deviation small enough. This is based on the well-known fact that within the range of 6 times standard deviation, a normal distribution covers 99.74% of data (Miller, 1990). This fact also allows applying normal distribution to EM_i and W_i, although both of them do have strict limited range. To make the experiments even more accurate, two other distributions have been used to model EM_i and W_i: triangular and beta distribution that have limited ranges. These additional distributions have not been applied to S as it has no upper limit defined.

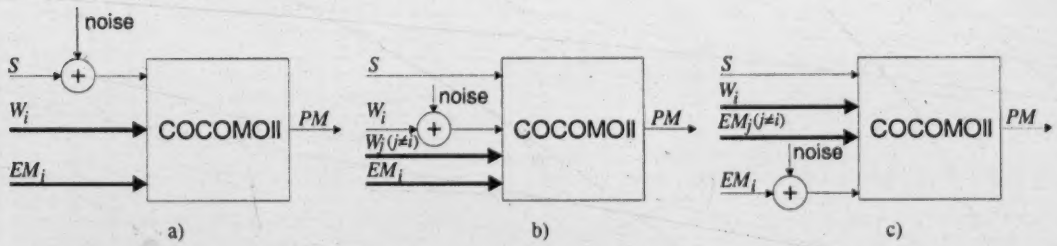


Figure 2: Monte Carlo simulation for single input error in: (a) size, (b) effort, (c), exponent scale factors

Each simulation run was performed using a random sample of N=1000 values generated randomly with given distribution. The standard deviation was set to 0.001 in the first run and gradually increased by 0.001 up to 0.3. The boundary value 0.3 was determined by examining the standard deviation to make the distributed data fall within valid range. The mean of the normal distribution was set to the actual values of S from the dataset. Mean

relative error *MRE* was determined for each simulation run. The relative error is defined here as the absolute difference of each simulation result *PM* and the mean value of all results in the run divided by this mean value. *MRE* is then the mean of all relative errors in a given run.

Table 19: Tolerable input errors found using Monte Carlo simulation

Distribution	Normal			Triangular		Beta	
Variable	S	W_i	EM_i	W_i	EM_i	W_i	EM_i
Project 1	5.10%	25.80%	6.20%	24.10%	5.80%	32.70%	10.10%
Project 2	5.10%	49.70%	6.40%	47.60%	6.10%	60.30%	10.90%
Project 3	5.10%	25.80%	6.10%	24.10%	6.00%	32.70%	9.80%
Project 4	5.10%	33.10%	6.30%	34.60%	6.00%	39.50%	10.70%
Project 5	5.20%	25.80%	6.10%	29.10%	6.00%	32.70%	9.80%
Project 6	5.10%	25.80%	6.10%	24.50%	6.00%	32.70%	9.80%
Project 7	5.20%	25.80%	6.10%	24.20%	6.10%	32.70%	9.80%
Project 8	5.10%	33.10%	6.10%	34.50%	6.00%	39.50%	9.80%
Project 9	5.10%	33.10%	6.30%	34.60%	6.00%	39.50%	10.70%
Project 10	5.20%	20.90%	6.10%	46.00%	6.10%	27.80%	9.80%
Project 11	5.10%	N/A	6.20%	N/A	5.80%	N/A	10.10%
Project 12	5.10%	N/A	6.20%	N/A	5.80%	N/A	10.10%
Project 13	5.20%	20.90%	6.10%	45.50%	6.00%	27.80%	9.80%
Project 14	5.10%	20.90%	6.30%	46.00%	6.00%	27.80%	10.70%
Project 15	5.20%	20.90%	6.20%	25.40%	5.80%	27.80%	10.10%
Average	5.13%	27.82%	6.19%	33.86%	5.97%	38.37%	10.13%

The mean relative errors are examined with the goal to find the largest input variable error corresponding to the mean relative error of the effort which is not greater than the threshold of 5% of the actual value of effort *PM*. This largest error, called tolerable input error, is a reflection of sensitivity. Tolerable input errors for given dataset and all three groups of parameters (EM_i, S and W_i) are shown in Table 19.

The table summarizes the tolerable input errors of S, W_i and EM_i found using three different distributions of input variables for all 15 sets of project data. Similarly to the results obtained by mathematical analysis, values of tolerable input error for W_i in projects 11 and 12 were not determined which means that the 5% of the effort error is never obtained even when W_i has input error of 100% of the actual W_i.

The bottom row of the Table 5 shows average values of tolerable input error for all three groups of parameters and the three random distributions considered. Overall, the results obtained by Monte Carlo simulation correspond very well to these obtained using mathematical analysis (see Table 4): the COCOMO II model is most sensitive to error in projected size S, followed by effort multipliers EM_i, with exponent scale factors W_i being the least important.

4.3.3 Error Propagation

Error propagation is applied in order to determine the impact of the cumulative input noise on the effort estimation. Shown in Figure 3, error propagation is a way of combining two or more random errors together to get a new variable representing the cumulative error. It is used in situations where it is not possible to measure the same parameter several times and thereby estimate the random error on the final result directly. It can be used when there is a need to measure more than one quantity to get the final result.

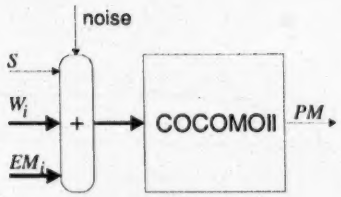


Figure 3: Error propagation

There are four basic rules (Mood, 1963, Agresti, 1990) to determine error propagation, assuming that the errors are normal in nature. In general it is assumed that a condition cannot be observed directly, but depends on other quantities that can be measured, e.g. x and y. The absolute uncertainties (standard deviation of Gaussian distribution) on these measurements are $\pm\delta x$ and $\pm\delta y$, respectively. The relative uncertainties of the data are then expressed as e.g. $\pm 100(\delta x/x)\%$. The rules can be summarized as follows:

Rule 1: The absolute uncertainty of $x+y$ or $x-y$ is obtained by adding the absolute uncertainties δx and δy in quadrature:

$$\delta(x-y) = \delta(x+y) = [(\delta x)^2 + (\delta y)^2]^{\frac{1}{2}} \tag{18}$$

Rule 2: The fractional uncertainty of x times y or x/y is obtained by adding the relative uncertainties δx and δy in quadrature.

$$\frac{\delta(xy)}{xy} = \frac{\delta\left(\frac{x}{y}\right)}{\frac{x}{y}} = \left[\left(\frac{\delta x}{x}\right)^2 + \left(\frac{\delta y}{y}\right)^2\right]^{\frac{1}{2}} \tag{19}$$

Rule 3: The relative uncertainty on x to a power n is just n times the relative uncertainty of x:

$$\frac{\delta(x^n)}{x^n} = n \times \frac{\delta x}{x} \tag{20}$$

In fact, this is true for any power of x, not just positive integer powers. Rules 1 and 2 depend on the assumption that the uncertainties on x and y have nothing to do with each other and may cancel each other out partially. This is clearly not true if we multiply a number by itself, which is why Rule 3 cannot be derived from Rule 2.

Rule 4: The absolute uncertainty on an arbitrary function of x is obtained by taking the derivative of that function, evaluating it at the value of the measured data, taking the absolute value, and multiplying the result by the absolute uncertainty on x. For clarity, we call the value of the measured data x_0.

$$\delta f(x_0) = \left|\frac{df}{dx}\right|_{x=x_0} \times \delta x_0 \tag{21}$$

Now using the rules above, Error propagation on PM is calculated assuming all the independent variables have the same noise and randomness.

Through observation of the COCOMOII equation, $PM = 2.94 \prod_{i=1}^{17} EM_i S^{0.91+0.01\sum_{i=1}^{5} W_i}$, several calculation steps are designed. It can be seen that there is a power expression in this equation, but Rule 3 cannot be applied because it requires the exponent to be a real

number, however, the exponent in COCOMOII equation is a random variable. As a solution to this, we apply natural logarithm to this equation to eliminate the random exponent

$$\log PM = \log \prod EM_i + 2.94 + (0.91 + 0.01 \sum W_i) \times \log S ,$$

express in terms of absolute uncertainty

$$\begin{aligned} \delta(\log PM) &= \delta[\log \prod EM_i + 2.94 + (0.91 + 0.01 \sum W_i) \times \log S] \\ &= \delta[\log \prod EM_i + \sum W_i \times \log S] \end{aligned}$$

and finally obtain

$$\delta(\log PM) = \delta(\log \prod EM_i + \sum W_i \times \log S). \tag{22}$$

Step 1, find $\delta[\Pi EM_i]$

Here rule 2 is used because it is simply a production. And it is easy to deduce that a fractional uncertainty of production of more than 2 random variables is also obtained by adding the fractional uncertainties of each input random variable.

$$\frac{\delta(\prod EM_i)}{\prod EM_i} = \left[\sum \left(\frac{\delta EM_i}{EM_i} \right)^2 \right]^{\frac{1}{2}} \tag{23}$$

Because $\delta EM_i = (L/1000) EM_i$ (the standard deviation of all 17 EMi is L/1000 times of itself, here L stands for the loop index when we increase the input error gradually), $\delta[\Pi EM_i] / \Pi EM_i$ is equal to $((L/1000)^2 \times 17)^{1/2}$, so from (23), $\delta[\Pi EM_i]$ is obtained,

$$\delta\left(\prod EM_i\right)=\prod EM_i\times\left[\left(\frac{L}{1000}\right)^{\frac{1}{2}}\times 17\right]^{\frac{1}{2}} \tag{24}$$

and here ΠEM_i stands for actual data value(so called measured value).

Step 2, find $\delta[\log\Pi EM_i]$

This is an arbitrary function, so rule 4 is applied. The derivative of $\log\Pi EM_i$ is $1/\Pi EM_i$, so, with (24) given,

$$\delta\left(\log\prod EM_i\right)=\frac{1}{\prod EM_i}\times\delta\left(\prod EM_i\right)=\left[\left(\frac{L}{1000}\right)^{2}\times 17\right]^{\frac{1}{2}} \tag{25}$$

Step 3, find $\delta[\sum W_i]$

Rule 1 is used here in that there are only summations.

$$\delta\left(\sum W_i\right)=\left(\sum(\delta W_i)^2\right)^{\frac{1}{2}}=\left[\sum\left(\frac{L}{1000}\times W_i\right)^2\right]^{\frac{1}{2}}=\frac{L}{1000}\times\left(\sum W_i^{\,2}\right)^{\frac{1}{2}}$$

$$\frac{\delta\left(\sum W_i\right)}{\sum W_i}=\frac{L}{1000}\times\frac{\left(\sum W_i^{\,2}\right)^{\frac{1}{2}}}{\sum W_i} \tag{26}$$

Step 4, find $\delta[\log S]/\log S$

Rule 4 is used here for the arbitrary function logS, the derivative of logS is 1/S, so

$\delta[\log S] = (1/S)\times(L/1000) \times S = l/1000$,

$$\frac{\delta(\log S)}{\log S} = \frac{L}{1000 \times \log S} \tag{27}$$

Step 5, find $\delta[\sum W_i \times \log S]$

Rule 2 is used, $\frac{\delta(\sum W_i \times \log S)}{\sum W_i \times \log S} = \left[\left(\frac{\delta \sum W_i}{\sum W_i}\right)^2 + \left(\frac{\delta \log S}{\log S}\right)^2\right]^{\frac{1}{2}}$

$$\frac{\delta(\sum W_i \times \log S)}{\sum W_i \times \log S} = \left[\left(\frac{\delta \sum W_i}{\sum W_i}\right)^2 + \left(\frac{\delta \log S}{\log S}\right)^2\right]^{\frac{1}{2}}$$

and with (26) and (27) given, $\frac{\delta(\sum W_i \times \log S)}{\sum W_i \times \log S} = \frac{L}{1000} \times \left[\frac{\sum W_i^2}{(\sum W_i)^2} + \frac{1}{(\log S)^2}\right]^{\frac{1}{2}}$

$$\delta(\sum W_i \times \log S) = \frac{L}{1000} \times \left[\sum W_i^2 \times (\log S)^2 + (\sum W_i)^2\right]^{\frac{1}{2}} \tag{28}$$

Step 6, find $\delta[\log \Pi EM_i + \sum W_i \times \log S]$

This step only needs rule 1, and with (25) and (28) given, the result is,

$$\delta(\log(\prod EM_i) + \sum W_i \times \log S) = \frac{L}{1000} \times \left[17 + \sum W_i^2 \times (\log S)^2 + (\sum W_i)^2\right]^{\frac{1}{2}} \tag{29}$$

(29) is also equal to δ[logPM].

Step 7, find δ[PM]

Because $PM = e^{logPM}$, with rule 4 applied,

$$\delta(PM) = e^{\log PM} \times \frac{L}{1000} \times \left[17 + \sum W_i^2 \times (\log S)^2 + \left(\sum W_i\right)^2\right]^{\frac{1}{2}}$$

$$= PM \times \frac{L}{1000} \times \left[17 + \sum W_i^2 \times (\log S)^2 + \left(\sum W_i\right)^2\right]^{\frac{1}{2}}$$

and the relative uncertainty on PM is

$$\frac{\delta(PM)}{PM} = \frac{L}{1000} \times \left[17 + \sum W_i^2 \times (\log S)^2 + \left(\sum W_i\right)^2\right]^{\frac{1}{2}} \quad (30)$$

where PM is calculated with actual data according the COCOMOII model and the variable *L* represents the level of uncertainty of the inputs to the model.

With equation (30), the dependent relative error can be determined when the input variables are added with increasing level of noise by changing *L*. Based on the first project data, with the standard deviations of all input variables set to *L*/1000 of the actual data value, the standard deviations (SD) of the dependent variable *PM* are listed in the 4 table 20.

Table 20: Standard deviations of PM for increasing levels of uncertainty L (Proj. 1)

L	1	2	3	4	5	6	7	8	9	10
SD	0.0722	0.1443	0.2165	0.2886	0.3608	0.4329	0.5051	0.5772	0.6494	0.7215

It can be seen that the error propagation is quite notable: when the input error for all independent variables are only 0.1% (1/1000 for L=1) of each of them, the error propagated on the dependent variable *PM* has reached to 7.22%. Therefore, for the preset tolerable error threshold equal to 5%, the input error should be confined within 0.1%.

The resulting values of tolerable input errors for all 15 projects are summarized in Table 21.

Table 21: Tolerable input error (TIE) obtained by error propagation

P	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TIE	0.1%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%

It is noticeable in that with exception of project 1, all the results have the same value.

In order to verify the result obtained by error propagation, another Monte Carlo simulation has been designed. Each variable is disturbed by noise with standard deviation starting from 1/1000 of its actual data and increasing by 1/1000 of the data in each step until the upper boundary is reached. The upper boundary of the standard deviation is subject to change with respect to different variables, because they have different defined ranges. The rest of the simulation is analogous to that described in Section 4.3.2. The tolerable input errors obtained by the Monte Carlo simulation for the threshold of 5% of the resulting effort estimation are listed in Table 22.

Table 22: Tolerable input error obtained by error propagation

Project	Normal Distribution	Normal + Triangular Distribution.	Normal + Beta Distribution.
1	0.30%	1.3%	4.2%
2	0.30%	1.4%	4.2%
3	0.30%	1.4%	4.2%
4	0.30%	1.4%	4.2%
5	0.30%	1.3%	4.2%
6	0.30%	1.3%	4.2%
7	0.30%	1.3%	4.2%
8	0.30%	1.3%	4.2%
9	0.30%	1.4%	4.2%
10	0.30%	1.4%	4.2%
11	0.30%	1.4%	4.2%
12	0.30%	1.4%	4.2%
13	0.30%	1.4%	4.2%
14	0.30%	1.4%	4.2%
15	0.30%	1.4%	4.2%

The results indicate that the tolerable input errors obtained with normal distribution based on Monte Carlo simulation have the smallest values, and that these values are comparable to the results obtained analytically. The tolerable input errors obtained with triangular and Beta distributions have much larger values caused by their different properties that fit the nature of effort multipliers and exponent scale factors. The tolerable input errors remain quite stable across different project data for the same distribution type, which means that the 15 data points have been collected in a consistent way.

4.4. Summary

Results obtained for single input noise cases by both mathematical analysis and Monte Carlo simulation consistently indicate that the most critical input variable is size S, closely followed by effort multipliers EM_i. The impact of error in determining exponent scale factors W_i is relatively small. This finding can be used to direct management in collecting and estimating these values: Size and effort multipliers are very important and adequate time and resources should be devoted to their accurate evaluation. The scale factors are much less important and could be neglected (set to their nominal values) if necessary.

The results obtained through error propagation are alarming showing very high sensitivity of the model when cumulative errors are considered. In such cases, a relatively small error in determining projected size of the system would lead to a large error of the estimated value of effort.

To cope with the problems of uncertainty in software cost estimation, we have proposed to use techniques such us fuzzy sets to represent variables entering the estimation process and relations between them. Two particular ways of incorporating fuzziness into cost estimation models are described in Sections 4 and 5.

5. Fuzzy Regression of COCOMO Model

The CoCoMo model, which is of our primary interest in this thesis is described by a log-linear function relating the size of the software project with the effort required for its completion. As it evolves into the newer CoCoMoII, it becomes more complicated and subject to many estimated values of cost drivers. This leads to questions about its feasibility, precision and efficiency, which this section is focused on.

Based on the available data sets gathered from four sources –USArmy, Yourdon, Belady-Lehman, and COCOMO63 database (refer to Appendix B), we build different models to capture the relationships among size/effort data. We start with linear regression, through robust regression, to fuzzy regression models. We take fuzziness (vagueness) into account in order to get our resulting model more practical, precise and reliable. Not only this allows explicit expression of uncertainty of the size but it also produces a fuzzy estimate of ensuing effort supporting designers' awareness of limited precision of the estimate.

Fuzzy sets extend the conventional notion of (binary) sets to fuzzy sets with continuously-valued membership. This extension allows for intermediate belongingness of a member in given set, besides the two extremes (yes/no) of crisp sets. More formally, a fuzzy set A is a mapping of a set of real numbers (xi) onto membership values (ui) in the range [0, 1]. This can be represented by a set of pairs ui/xi, where ui is the membership value for the real number xi, i.e. {u1/x1, u2/x2 ... un/xn} with values of x in

increasing order (x1 < x2 < ... < xn). Values prior to x1 have the same membership value as x1 and values after xn have the same membership value as xn. Values between xi and xi+1 are determined by the value that lies on the straight line between the 2 consecutive points. In effect straight lines can be used to join the points in the fuzzy set and horizontal lines to extend to the first and from the last points. A very simple type of fuzzy set, triangular fuzzy set is extensively used in this thesis and there are many more other kind of fuzzy sets with different shapes.

5.1. Fuzzy linear Regression

All the data this experiment is based on are two dimensional, indicating the size (in KLOC) and effort (in person month) of the software projects. By combining the four sources of data, 128 data points are obtained, which correspond to 128 software projects

First the experiment needs to find the non-fuzzy linear relation between the size and effort using Robust linear regression techniques, and this results in a linear function computing effort from size with two coefficients. Then fuzziness is added to the two coefficients (resulting from the Robust linear regression), the values of the two coefficients are set as the modal values of the fuzzy numbers. For simplicity consideration, triangular fuzzy number is chosen.

To find out the span (upper and lower boundaries) of the two fuzzy numbers, the problem is put into a linear programming scheme where the objective function is to minimize the

fuzziness of the triangular fuzzy numbers and the constraints are to let our output fuzzy numbers "cover" as many data points as possible.

5.1.1 Least Square Linear regression and robust regression

Both based on smallest least square method, Least Square Linear regression and robust linear regression only differ in one thing, robust regression neglects the impact of the outliers (those points too scattered). Figure 4 shows the two resulting regression lines. The upper one is Least Square regression line (with circles), the lower is the robust regression line, and the crosses stand for the original data points.

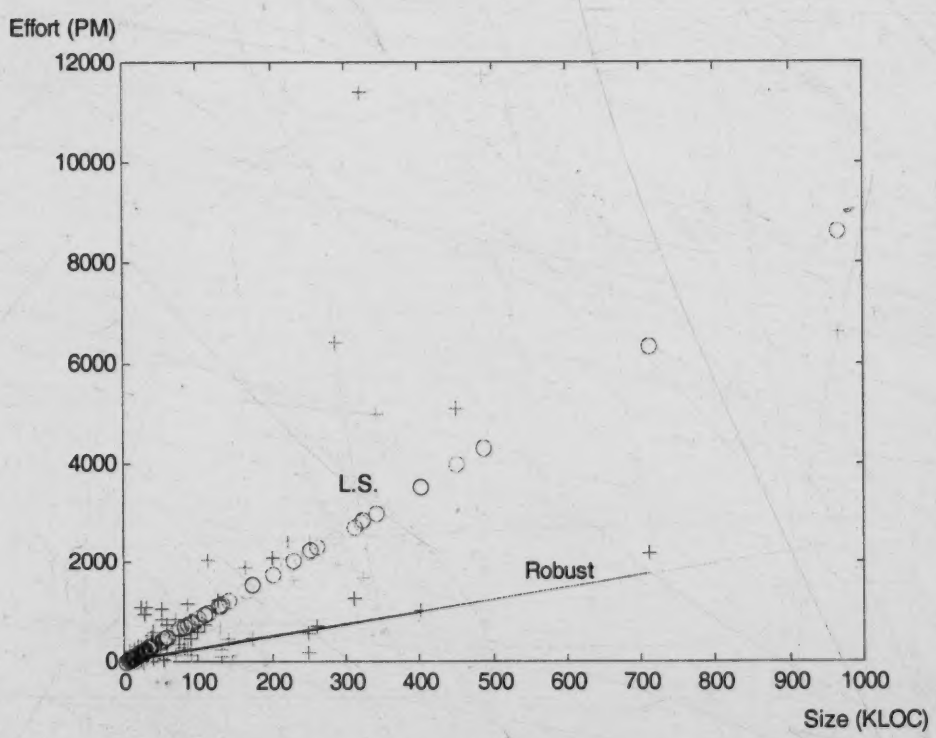


Figure 4: Linear regression and Robust regression

Since most of the points lies where size is much smaller than 1000 KLOC. Figure 5 is a dilated closer look of the regressions

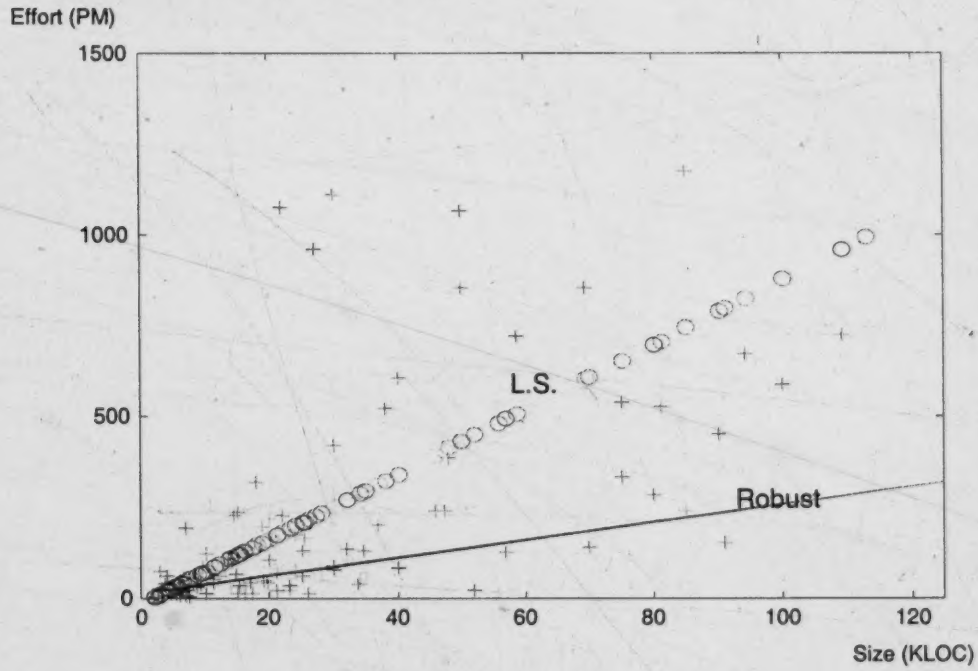


Figure 5: Linear and Robust regression-closer look

The Least Square regression equation is

$$Effort = 8.9Size - 12.4 \tag{31}$$

with standard errors for intercept (-12.4) to be 130.0310 and size to be 0.8111. However, Robust regression gives a better performance with much less standard error with intercept (7.5758) and size (0.1052), and the equation for Robust regression is

$$Effort = 2.5Size + 13.4 \tag{32}$$

So we believe the Robust regression result is better, let E stand for Effort, K stand for size, the relation between size K and effort E is:

$$E = 2.5K + 13.4 \tag{33}$$

5.1.2 Fuzzified model

Let triangular fuzzy number A, B replace 2.5 and 13.4 respectively. The lower boundary, modal value and upper boundary of A, B are [a1, 2.5, b1] and [a2, 13.4, b2] respectively. Tiangular fuzzy number A multiplied by a crisp number K, then added by another triangular fuzzy number B, is still a triangular fuzzy number [a1*K+a2, 2.5*K+13.4, b1*K+b2].

5.1.3 Linear programming

A α value (0.9) is set to let the corresponding α-cut cover all the data. Therefore, for each given Ki (size), the corresponding Ei (effort) is lower than the upper boundary of the α-cut, and upper to the lower boundary of the α-cut. That is,

$$\begin{cases} E_i \geq C_i - (C_i - A_i)(1-\alpha) \\ E_i \leq C_i + (B_i - C_i)(1-\alpha) \\ A_i = a_1 K_i + a_2 \\ B_i = b_2 K_i + b_2 \\ C_i = 2.5K_i + 13.4 \end{cases} \tag{34}$$

Figure 6 gives an intuitive description of this scenario. The central line covering C_i is the modal values of all the fuzzy sets (it's also the result of the linear regression based on the data). The upper and lower lines stand for the upper bound and the lower bound of the fuzzy sets respectively. There is a dashed line triangular standing for the triangular fuzzy set.

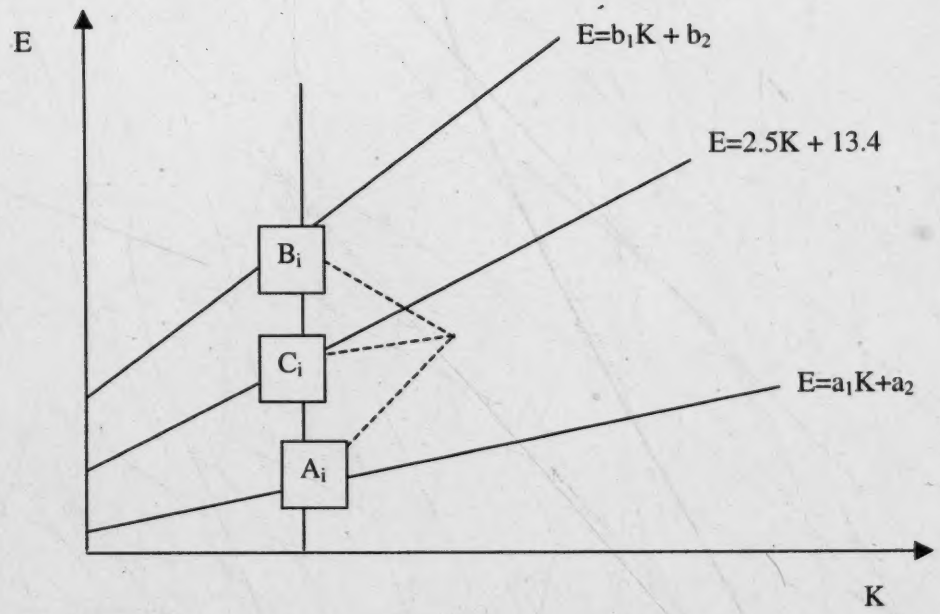


Figure 6: Fuzzy regression

From equation (34), equations (35) and (36) can be deduced.

$$a_1 K_i + a_2 \leq 10E_i - 9(2.5K_i + 13.4), \quad i = 1,2,3...128 \tag{35}$$

$$b_1 K_i + b_2 \geq 10E_i - 9(2.5K_i + 13.4), \quad i = 1,2,3...128 \tag{36}$$

The values of the points (K_i, E_i) are available in the data set. So above are the linear constraints with variables a_1, a_2, b_1 , b_2.

The objective function is set to minimize the fuzziness of the triangular fuzzy number.

$$\min \textit{fuzziness} = \frac{(B_i - A_i)\times 1}{2C_i} \Rightarrow \min \frac{B_i - A_i}{C_i}$$

$$\min \frac{K_i(b_1 - a_1) + (b_2 - a_2)}{2.5K_i + 13.4}$$

For each given K_i is a constant, hence

$$\min(b_1 - a_1) + b_2 - a_2 \Rightarrow \min b_1 + b_2 - a_1 - a_2$$

As such in summary, the linear programming problem can be solved by the functions below:

The objective function is,

$$\textbf{min}\ (b_1 + b_2 - a_1 - a_2) \tag{37}$$

The constraint inequalities are,

$$a_1 K_i + a_2 \le 10E_i - 9(2.5K_i + 13.4), i = 1,2,3,....128 \tag{38}$$

$$b_1 K_i + b_2 \ge 10E_i - 9(2.5K_i + 13.4), i = 1,2,3,....128 \tag{39}$$

Using BPMPD Version 2.21 to execute the linear programming problem and the result is:

a_1 = 1.9e-13, a_2 = - 9.5e-10 ,
b_1 = 460 , b_2 = 13.4

That is, the resulting fuzzy coefficients are:

A = [1.9e-013, 2.5, 460]
B = [-9.5e-10, 13.4, 13.4]

Below is Figure 7 which encompasses area (by the two lines) of all data, and the lower line seems non-increasing because of the large scale of the y-coordinate.

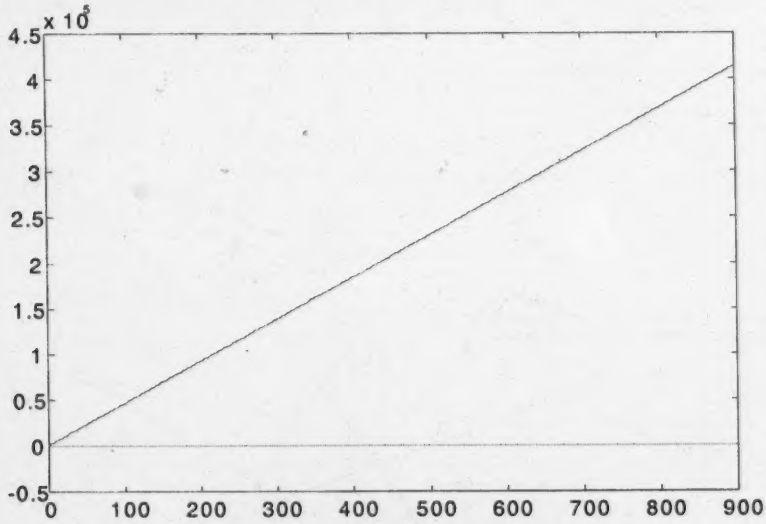


Figure 7: Linear regression

To improve the scattered figure, the data set is then restricted to only COCOMO81 data set that includes only 61 points with two most scattered points taken out (it contains originally 63 points). From the Figure 8 we can see a much improved new upper bound line. The lower bound is not much changed and is almost overlapped by the previous one. However, the upper bound is much more restricted to a lower position.

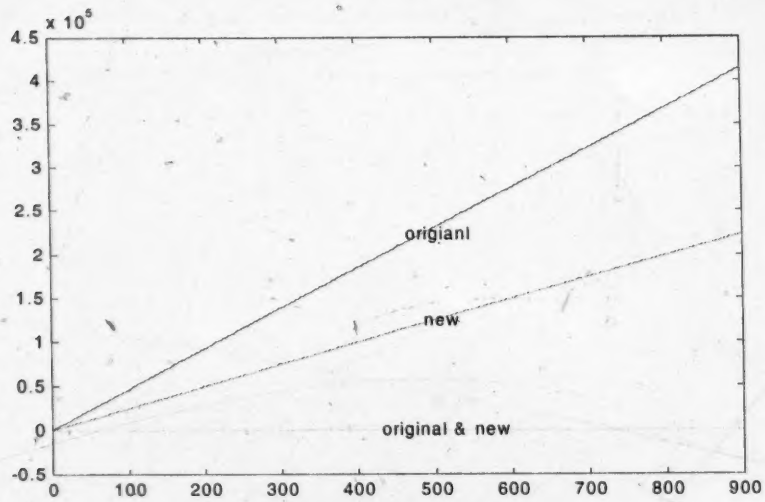


Figure 8: improved linear regression

The equation for new upper line is,

$$E = 248K \tag{40}$$

The equation for the new lower line is,

$$E = 1.154e(-11) \times K + 0.954 \tag{41}$$

5.2. Polynomial fuzzy regression

The model to be considered toward a polynomial regression is

$$E = iK^2 + jK + t \tag{42}$$

In which i, j, t are triangular fuzzy numbers [a1, i, b1], [a2, j, b2], [a3, t, b3]. To improve the consistence of the data, only CoCoMo81 data are used, and also only 61 of all 63 data points are considered (two outlying points eliminated).

The resulting fuzzy number E is not triangular any more, but its modal value and boundaries can still be described by a tripple [Ai, Ci, Bi] where,

$$\begin{cases} A_i = a_1 K_i^{\,2} + a_2 K_i + a_3 \\ B_i = b_1 K_i^{\,2} + b_2 K_i + b_3 \\ C_i = i K_i^{\,2} + j K_i + t \end{cases} \tag{43}$$

Again linear programming is used to guarantee that all E_i 's lie between their corresponding A_i 's and B_i's,

$$A_i \le E_i \le B_i,$$

That is,

$E_i \le B_i$ $(E_i \le b_1 K_i^{\,2} + b_2 K_i + b_3)$

$E_i \ge A_i$, $(E_i \ge a_1 K_i^{\,2} + a_2 K_i + a_3)$, and that $\frac{B_i - A_i}{C_i}$ is minimized (smallest fuzziness), as shown in Figure 9.

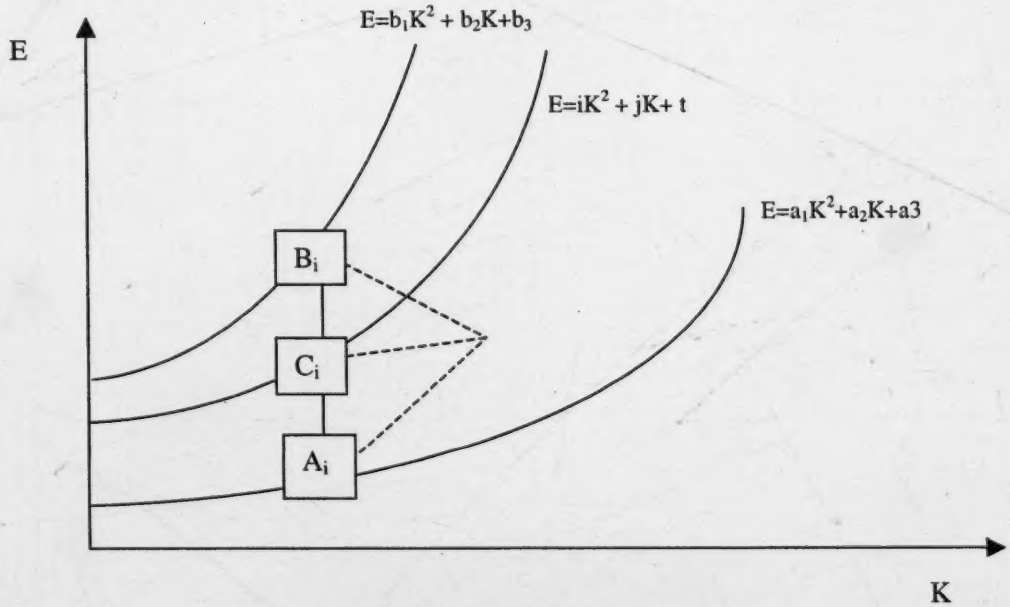


Figure 9: Polynomial fuzzy regression

Hence the objective function will be,

$$\textbf{min } b_1 + b_2 + b_3 - a_1 - a_2 - a_3$$

And the constraints are,

$$a_1 K_i^{\,2} + a_2 K_i + a3 \le E_i \quad , \quad i = 1,2,....61 \tag{44}$$

$$b_1 K_i^{\,2} + b_2 K_i + b3 \ge E_i \quad , \quad i = 1,2,....61 \tag{45}$$

Again BPMPD Version 2.21 is used to get the fuzzy polynomial regression:

$$E = [0, 0.0119, 1.8661]\, K^2 + [0, 4.859, 4.859]\, K + [5.9, 65.089, 65.089]$$

As show in Figure 10.

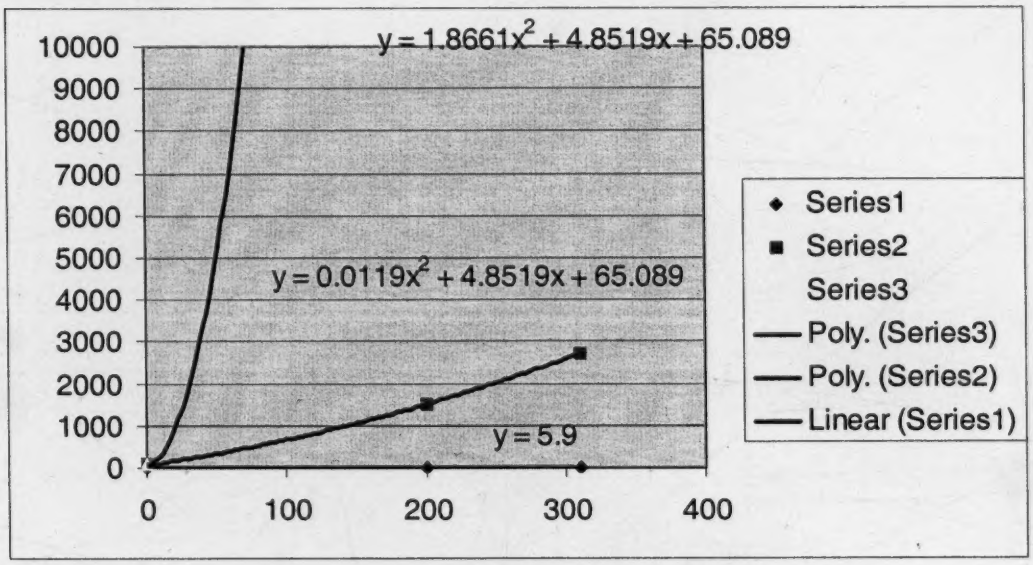


Figure 10: Polynomial regression result

It can be seen that the upper bound line starts to increase too fast at the very beginning, which results in a very scattered model.

5.3. Summary

Both linear and polynomial fuzzy regressions result in very scattered models. In reality, such model won't be practical at all. This is because the regressions are based on very strict linear programming mechanism, i.e. all data must lie within the area defined by the upper and lower boundary. Therefore, fuzzy regression approach cannot be directly used to enhance software estimation models as the available data are highly inconsistent and lead to too wide fuzzy regression models with no practical meaning. Another approach, considering fuzzy numbers and fuzzy arithmetic, is described in Section 6.

6. Fuzzy Extension of COCOMO model

6.1. Sources of uncertainty in the cost estimation models

Traditionally, the problem of software cost estimation relies on a single (numeric) value of size for given software project to predict the ensuing effort or cost. However, the size of the code is, especially at the beginning of the project, a matter of estimation (e.g., based on some previously completed projects that resemble the current one). Obviously, correctness and precision of such estimates are limited. It is of paramount importance to recognize this situation and come up with a technology using which we can evaluate the associated imprecision residing within the final results of cost estimation [Musilek, 2001]. The technology endorsed here deals with fuzzy sets.

Using fuzzy sets, size of a software project can be specified by distribution of its possible values. Commonly, this form of distribution is represented in the form of a fuzzy set. For example, a small software project can be described by a fuzzy set K in the form shown in Figure 11. What becomes apparent here is a continuous character of belongingness (membership) of elements to the given concept (being here a small software project). In this sense, fuzzy sets help alleviate a dichotomy problem (yes-no evaluation) that could be very artificial to grasp using the language of set theory. The grades of membership capture a notion of partial membership of an element to the concept (fuzzy set).

In general, a fuzzy set K is described by its membership function $K(x)$ which expresses the degree of membership of x to the fuzzy set K describing a certain concept (say, small project, high reliability, etc.).

6.2. Fuzzy Numbers

Fuzzy sets defined in **R**, or *fuzzy numbers* (FN), can easily represent various concepts with partial membership. The membership grades can be captured through membership functions. There are several basic models of membership functions of fuzzy numbers such as triangular, parabolic, Gaussian, to name the most commonly used. Generally, membership functions may not be symmetric. Quite often this is the case as we may anticipate a certain tendency as to the estimation of the size of the code. From this perspective, triangular and parabolic fuzzy sets are of special interest in our thesis.

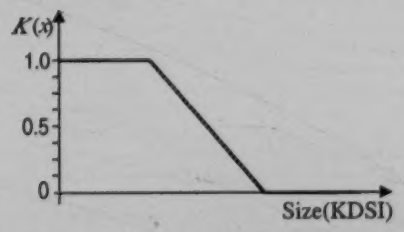


Figure 11: the size of a small software project modeled as a fuzzy set

6.2.1 Triangular Fuzzy Numbers

A triangular fuzzy number (TFN) K, see Figure 12, is described by a triplet $\{\alpha, m, \beta\}$ where m is the modal value of the fuzzy number K, and α and β are its left and right boundary values, respectively

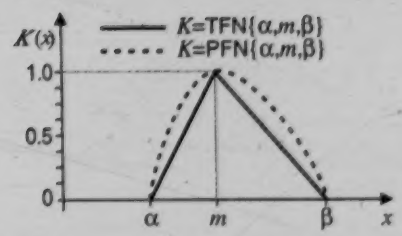


Figure 12: Triangular and parabolic fuzzy numbers

The membership function of the triangular fuzzy number K is defined in the form

$$K(x) = \begin{cases} \dfrac{x-\alpha}{m-\alpha} \text{ for } x \in [\alpha, m] \\ \dfrac{\beta-x}{\beta-m} \text{ for } x \in [m, \beta] \\ 0 \quad \text{for } x \notin [\alpha, \beta] \end{cases} \tag{46}$$

In the sequel, we will be using a shorthand notation TFN{α, m, β}that explicitly summarizes the parameters describing this fuzzy number.

6.2.2 Parabolic Fuzzy Numbers

Parabolic fuzzy number (PFN) is composed of two parabolic segments of the membership grades delimited by modal and boundary values. It is described by a triplet TFN{α, m, β}with modal value m, and parameters of the left and right boundaries α and β.

The membership function of the parabolic fuzzy number K is defined in the form

$$K(x)=\begin{cases}1-\dfrac{(x-\alpha)^2}{(m-\alpha)^2} & \text{for } x\in[\alpha,m]\\ 1-\dfrac{(\beta-x)^2}{(\beta-m)^2} & \text{for } x\in[m,\beta]\\ 0 & \text{for } x\notin[\alpha,\beta]\end{cases} \tag{47}$$

see also Figure 12.

6.2.3 Fuzziness of Fuzzy Numbers

Fuzzy numbers are special fuzzy sets representing uncertain quantitative information. They are convex and normal, usually with single modal value. They are also associated with some vagueness or *fuzziness*. For the evaluation of the uncertainty of fuzzy quantities in software cost estimation problem, we use measure of relative fuzziness f defined as

$$f(A)=\frac{\int_{\alpha}^{\beta}A(x)\mathrm{d}x}{\mathrm{m}}, \tag{48}$$

where A is a fuzzy number with spread $[\alpha,\beta]$. The higher the value of $f(A)$, the more "uncertain" A is.

6.2.4 Fuzzy Arithmetic

Computations involving fuzzy numbers are carried out in the setting of so-called fuzzy arithmetic (Dubois, 1978)(Mares, 1994). It dwells on the extension principle (Zadeh, 1975). In a nutshell, given a function (mapping) f such that f: **R** -> **R**, the extension principle deals with the generalization of this mapping to the case of arguments being fuzzy numbers that is $B=f(A)$, where A and B are fuzzy numbers.

For instance consider that $C = f(A, B)$. Then the membership function of C is computed as

$$C(z) = \sup_{x,y \in R: z = f(x,y)} [\min(A(x), B(y))] \tag{49}$$

Essentially, the above is nothing but a problem of nonlinear programming

$$\sup_{x,y \in R} [\min(A(x), B(y))] \tag{50}$$

with constraints represented by the mapping f under consideration

$$z = f(x, y). \tag{51}$$

The extension principle generalizes to mappings involving any number of the variables.

6.3. Simple f-COCOMO Model

Let us pose the simplest case of granular COCOMO model considering only size of the project K to be fuzzy while the coefficients a and b are crisp. We will call this model Simple f-COCOMO model. As the input variable (size K) is a fuzzy set (fuzzy number), so is the effort E.

The determination of the membership function of effort is based on the extension principle as shown in the previous section. More specifically, we get

$$E(e) = \sup_{x \in R: e = ax^b} [K(x)] \tag{52}$$

where $K(x)$ and $E(e)$ denote membership functions of the size of the code and membership function of the effort, respectively. We rewrite (8) in a different format by eliminating the constraints and moving them directly into the membership function (note that in this problem the constraint is a one-to-one mapping). We get

$$E(e) = \sup_{x \in \mathbf{R}: e = ax^b} [K(x)] = K((\frac{e}{a})^{1/b}). \quad (53)$$

In the following, we show detailed computation of effort in simple f-COCOMO model for two types of fuzzy numbers.

6.3.1 Simple f-COCOMO Model: Triangular Numbers

For triangular fuzzy numbers, we substitute (46) for $K(x)$ in (53) to get (54).

This formula can be used directly for estimation of effort from size of software project described by triangular fuzzy set. The results of TFN based simple f-COCOMO model for a selected software project are summarized in Figure 13.

$$E(e) = \begin{cases} \dfrac{\left(\frac{e}{a}\right)^{1/b} - \alpha}{m - \alpha} & \text{for } e \in [a\alpha^b, am^b] \\ \dfrac{\beta - \left(\frac{e}{a}\right)^{1/b}}{\beta - m} & \text{for } e \in [am^b, a\beta^b] \\ 0 & \text{for } e \notin [a\alpha^b, a\beta^b] \end{cases} \quad (54)$$

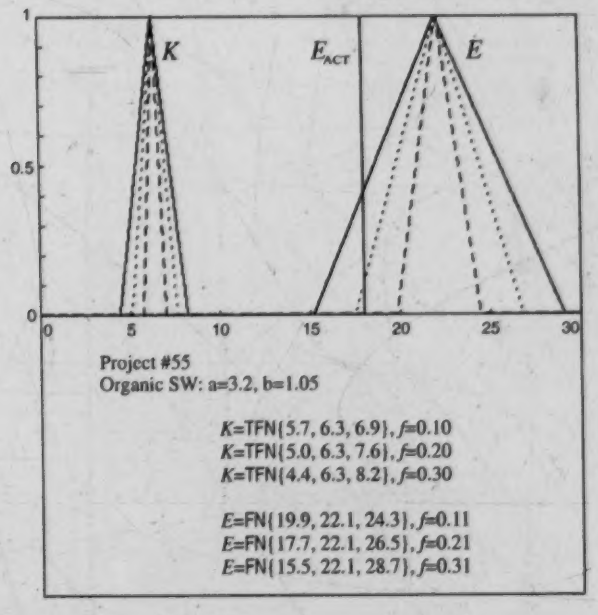


Figure 13: Simple f-COCOMO model with triangular fuzzy numbers

6.3.2 Simple f-COCOMO Model: Parabolic Numbers

For parabolic fuzzy numbers, we substitute (47) for $K(x)$ in (53) and get

$$E(e)=\begin{cases} 1-\dfrac{\left[\left(\dfrac{e}{a}\right)^{1/b}-m\right]^2}{(m-\alpha)^2} & \text{for } e\in\left[a\alpha^b, am^b\right] \\ 1-\dfrac{\left[\left(\dfrac{e}{a}\right)^{1/b}-m\right]^2}{(\beta-m)^2} & \text{for } e\in\left[am^b, a\beta^b\right] \\ 0 & \text{for } e\notin\left[a\alpha^b, a\beta^b\right] \end{cases} \tag{55}$$

The obtained formula can be used directly for estimation of effort from size of software project described by parabolic fuzzy set.

The relationships (54) and (55) quantify the way in which propagation of uncertainty (granularity) occurring at the input of the model is realized. It is important to stress that uncertainty at the input level of the COCOMO model yields uncertainty at the output. This becomes obvious and, more importantly, bears a substantial significance in any practical endeavor. By changing the size of the fuzzy set of size of the code (that reflects a level of designer's confidence as to the estimate), we can easily model how such estimate impacts the effort. Obviously, a certain monotony property holds, that is, less precise estimates of size give rise to less detailed effort estimates (cf. again Figure 12).

In addition, different categories of software require the use of different values of coefficients a and b. This also leads to different fuzziness of the estimated effort E. Series of plots in Figure 14 illustrate this effect.

It is evident that f-COCOMO model provides estimates of software development effort in terms of possibility distributions described by fuzzy numbers E. Such form of results is more confined when compared to a single numerical value obtained using common COCOMO model. In other words, even if the mean value of the fuzzy estimate E does not correspond to the actual value of development effort E_{ACT}, it may be covered by the estimate as possible with degree $E \in (0,1)$. This phenomenon is illustrated in Figure 15 by series of histograms of possibility values of E_{ACT}, as estimated by the f-COCOMO model for various values of spread of the input fuzzy numbers K. The relation between the value of spread (and, therefore fuzziness f) of K and coverage of the actual values E_{ACT} is shown in Figure 16. The two curves depict relative number of nonzero possibilities and average possibility of E_{ACT} for each value of spread $\delta = (\beta - \alpha)/\tilde{m}$.

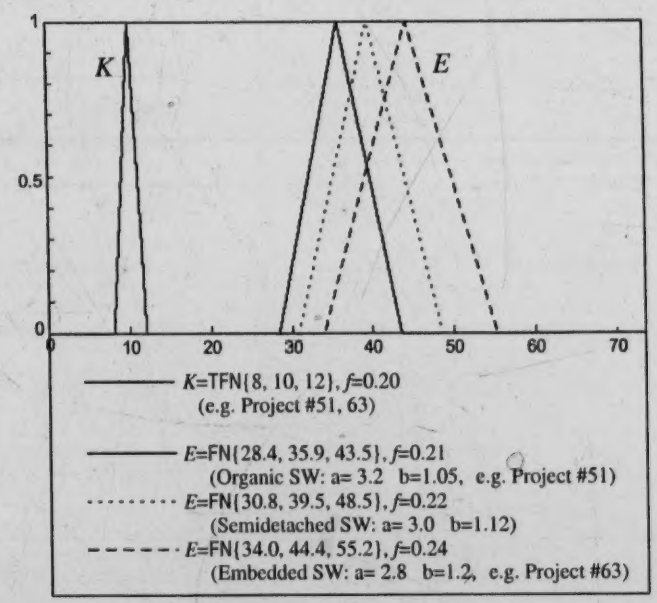

K
E
K=TFN{8, 10, 12}, f=0.20
(e.g. Project #51, 63)
E=FN{28.4, 35.9, 43.5}, f=0.21
(Organic SW: a= 3.2 b=1.05, e.g. Project #51)
E=FN{30.8, 39.5, 48.5}, f=0.22
(Semidetached SW: a= 3.0 b=1.12)
E=FN{34.0, 44.4, 55.2}, f=0.24
(Embedded SW: a= 2.8 b=1.2, e.g. Project #63)

Figure 14: Simple f-COCOMO model with TFN for different development modes

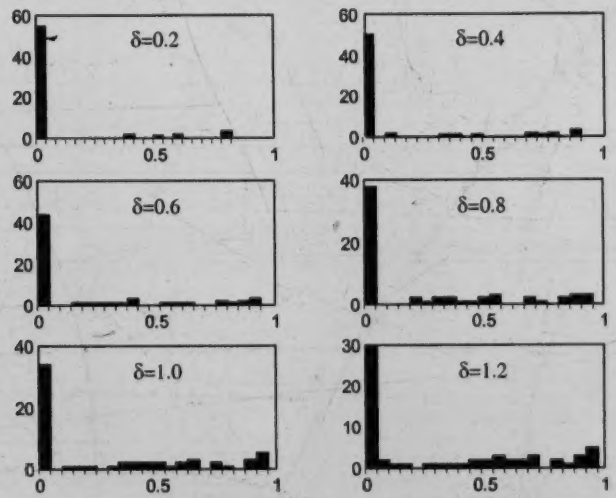


Figure 15: Histograms of possibilities of EACT

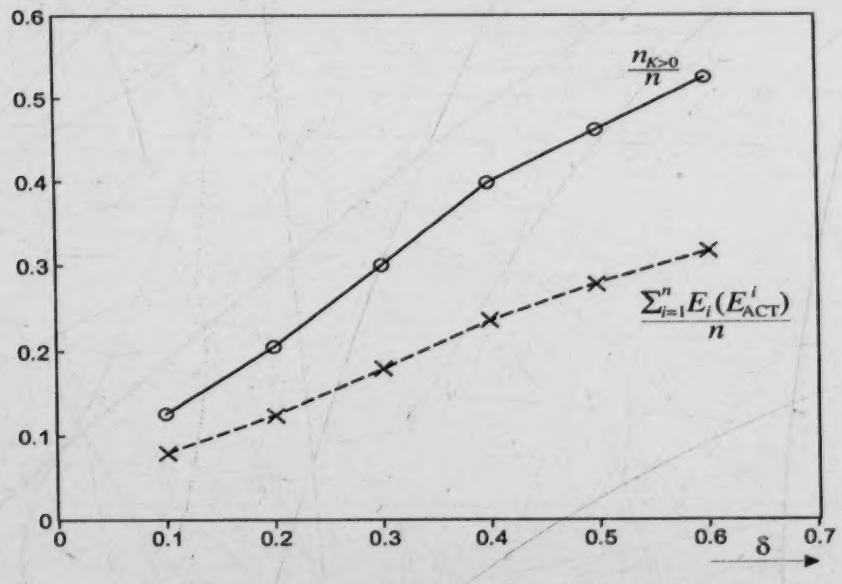


Figure 16: Spread δ vs. coverage of EACT

Similar experiments have been run among different groups of software projects to explore potential contrasts in suitability of the proposed method for some of them. The results obtained for δ=1 are summarized in Table 23. The data in parentheses indicate the number of cases for each group.

Table 23: Average value of possibility measures obtained using simple f-COCOMO model for various categories of software (CAT) and application domains (APP).

	MODE		ORGANIC		SEMI DETACHED		EMBEDDED	
TYPE	(63)		0.14	(23)	0.32	(10)	0.24	(30)
BUSINESS	0.21	(7)	0.28	(4)	0	(1)	0.18	(2)
CONTROL	0.23	(10)	0	(0)	0	(1)	0.26	(9)
HUMAN MACHINE	0.44	(13)	0	(1)	0.48	(2)	0.48	(10)
SCIENTIFIC	0.21	(17)	0.21	(11)	0.40	(2)	0.11	(4)
SUPPORT	0.27	(8)	0.32	(4)	0.17	(3)	0.37	(1)
SYSTEMS	0.31	(8)	0.22	(3)	0.88	(1)	0.24	(4)

6.4. Augmented f-COCOMO Model

An interesting generalization arises when we admit that the software project may concern a system whose membership to one of the three system categories is not obvious. Then we may view this as a fuzzy set and view this as such in the above calculations. For instance, the fuzzy set in Figure 17 concerns the system that is predominantly embedded (with membership degree equal to 0.9) with some far lower membership (0.3) in the category of semidetached software. In this sense, the coefficient a is a fuzzy set with discrete membership function of the form $A = \{0.9, 0.3, 0\}$. Similarly, we treat the second parameter b as a discrete fuzzy set B equal to $\{0.9, 0.3, 0\}$.

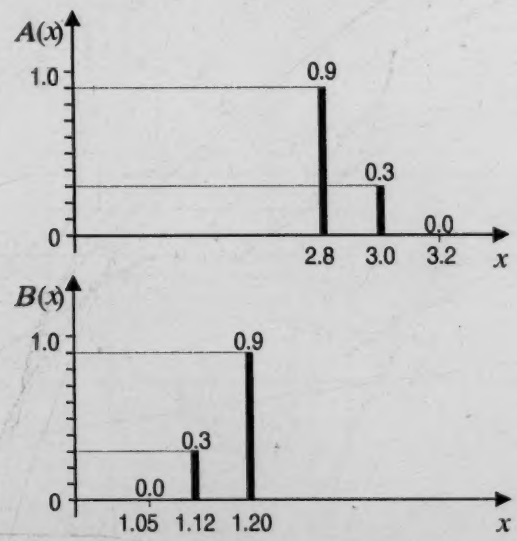


Figure 17: Parameters a and b as discrete fuzzy numbers

Taking this into consideration, we use the extension principle treating now K, a, and b as fuzzy sets (fuzzy numbers). More specifically, the calculations are carried out in the form

$$E(e) = \sup_{x,a,b \in \mathbf{R}, e=ax^b} [\min(K(x), A(a), B(b))] \quad (56)$$

This leads to relation

$$E(e) = \max_{a,b \in \mathbf{R}} \left\{ \min \left[K\left((e/a)^{1/b} \right), A(a), B(b) \right] \right\} \tag{57}$$

that can be directly used to determine the effort estimate E. As fuzzy numbers A and B are discrete fuzzy sets, it can be further simplified to the form

$$E(e) = \max_{\substack{a \in \{2.8, 3.0, 3.2\} \\ b \in \{1.2, 1.12, 1.05\}}} \left\{ \min \left[K\left((e/a)^{1/b} \right), A(a), B(b) \right] \right\} \tag{58}$$

The results of the augmented f-COCOMO model applied to a project with K={8,10,12} and development mode parameters A = B={0.9, 0.3, 0} (cf. Figure 10) are shown in Figure 18.

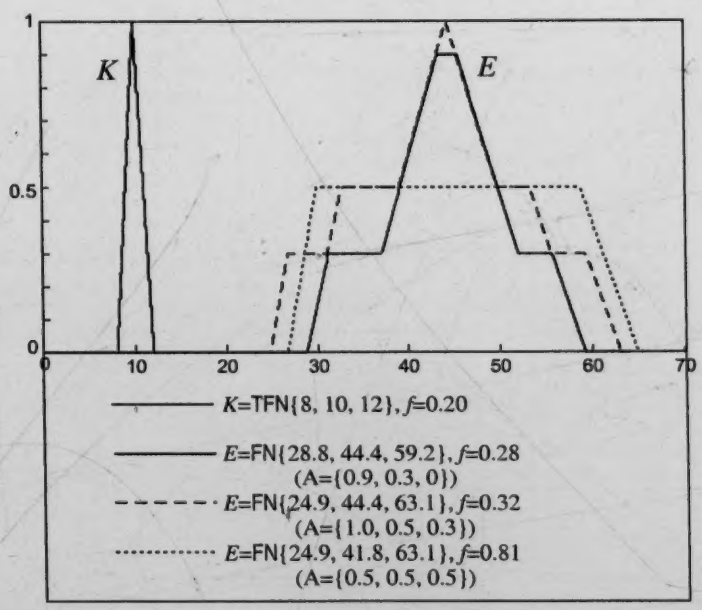


Figure 18:Augmented f-COCOMO model with triangular fuzzy numbers

The figure shows also two other interesting examples with development mode parameters equal to {1, 0.5, 0.3}, and {0.5, 0.5, 0.5}. This series of results also demonstrates the growth of uncertainty of resulting estimate of effort E with growing vagueness of determination of the development modes.

6.5. Summary

In this section, we have introduced an important generalization of the COCOMO software cost estimation model by augmenting by the technology of fuzzy sets. The detailed formulas involving several main classes of fuzzy numbers were derived and a set of intensive experiments provided a detailed insight into the conceptual and algorithmic essence of this generalization.

One should stress that the f-COCOMO model supports an important aspects of the "*what - if*" analysis (Pedrycz, 1998). The model admits inputs (say, size of code) that are non-numeric and therefore more acceptable to the designer and project manager. Fuzzy sets (as opposed to standard interval analysis) create a more flexible, highly versatile development environment. First, they help articulate the estimates and their essence (e.g., by exploiting fuzzy numbers described by asymmetric membership functions). Second, they generate a feedback as to the resulting uncertainty (granularity) of the results. The decision-maker is no longer left with a single number estimate which could be highly misleading in many cases and lead to the belief as to the relevance of the obtained results.

Moreover, by capturing the uncertainty of the initial data (estimates), one can monitor the behavior (quality) of the cost estimates over the course of the software project. This facet adds up a new conceptual dimension to the models of software cost estimation by raising awareness of the decision making with regard to the quality of the initial data needed by the model.

The methodology of fuzzy sets giving rise to the f-COCOMO extension is sufficiently general to be applied to other models of software cost estimation (such as the well-known function point method (Matson, 1994) and to other areas of quantitative software engineering.

7. Conclusion

Software cost estimation is an important activity of software engineering and project management in general. It is used throughout the software development lifecycle to provide vital information on project feasibility and resources needed for project completion. Therefore, availability of a general and reliable cost estimation model is highly desirable. The existing software cost/effort estimation methods provide a broad range of views on software engineering economics by considering economies of scale and various effort adjustment factors along with size- or function-based description of the software projects. Although some of them can be successfully applied to a certain class of software products, no single model provides reliable estimates across different software projects or different software organizations.

In this thesis, two major and popular software cost estimation models are considered and analysed from the point of view of their sensitivity to imprecision inherent in estimates of the parameters of these model. To provide a rigorous view of the model's sensitivity several methods were employed, including mathematical analysis, Monte Carlo simulation and error propagation analysis. The results of sensitivity analysis show quite alarming results, especially when considering multiple imprecision in several of the estimated parameters.

To overcome the problem of imprecision, that renders performance of the model practically unusable, techniques of fuzzy modelling were employed to enhance the model

into the domain of fuzzy quantities. It was expected that, by explicitly acknowledging the imprecision, the resulting models provide more accurate estimates accompanied by a measure of belief of their accuracy.

The first approach, consisting in development of fuzzy regression model for cost estimation, showed to be impractical due to very large spread of the estimates. In other words, fuzzy regression model provides estimates with too low belief measure.

The second approach incorporated the notion of fuzzy quantities and fuzzy arithmetic. In this case, the parameters entering the model (e.g. size) are considered to be fuzzy thus explicitly reflecting the modeller's uncertainty about their values. The output of this extended model, called fuzzy-COCOMO, comes also in form of a fuzzy number carrying both the modal value of the estimated effort and a belief measure described implicitly by the number's fuzziness (spread). In this case, the spread of the estimates is reasonable and provides valuable add-on information regarding reliability and usability of the estimated values.

In a brief summary, the contribution of this thesis can be summarized in the following points:

- Providing and overview of all major software cost estimation models, putting them in a historical context, and showing their mutual links and relative merits.
- Rigorous analysis of sensitivity of two major and popular cost estimation models using three different analytical and simulation methods.

- Introduction of the technology of fuzzy sets into the problem of software cost estimation.
- Development of fuzzy regression model and its critical analysis showing its shortcomings for practical use.
- Development of several fuzzy arithmetic models, including basic and enhanced fuzzy-COCOMO models, and their critical review demonstrating its great potential for application in cost estimation practice.

Even though fuzzy-COCOMO is a functional prototype, there are some areas in which it can be extended. In order to extend its practical value, a graphical user interface should be developed allowing the user to perform a valuable what-if analysis by manipulating a graphical representation of fuzzy quantities describing the model using a mouse pointer. Also fuzziness of the estimates should be interpreted more directly in terms of their further usability in process management, e.g. by expressing it as a percentage or a normalized coefficient.

8. References

Agresti, A. (1990), Categorical Data Analysis, Wiley, New York,

Albrecht, A. J. (1979). Measuring Application Development Productivity, Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Oct. 1417, 1979

B. W. Boehm et al. (2000). Software Cost Estimation with COCOMO II, Prentice Hall

Basili, V. R., K. Freburger. (1981). Programming Measurement and Estimation in the Software Engineering Laboratory, Journal of Systems and Software, 2, 2(1981), 47-57

BIS/Estimator User manual version 4.4 BIS Applied System Ltd. (1987).

Boehm, B. (1999), An Overview of the COCOMO 2.0 Software Cost Model.

Boehm, B. W. (1981). Software Engineering Economics, Prentice Hall.

Boehm, B.W. (1981). Software Engineering Economics. Prentice Hall.

Boehm, B.W., Abts, C., Clark, B., and Devnani-Chulani. S. (1997). COCOMO II Model Definition Manual. The University of Southern California

Briand, L. C., K. El Emam, I. Wieczorek (1999). Explaining the Cost of European Space and Military Projects, ICSE' 99: Proceedings of the International Conference on Software Engineering, Los Angeles, CA, ACM, 303-312.

Christensen, R. A. (1989), Data Distributions: a Statistical Handbook. Entropy Limited, Lincoln, Mass.

Computer Associates CA-Estimacs User Guide Release 5.0 (1986).

Dubois, D., H. Prade (1978). Fuzzy real algebra - some results, Fuzzy Sets and Systems, 2, 2, 327-348.

Fenton, N.E. and Pfleeger, S.L., (1997). Software Metrics: A Rigorous and Practical Approach. International Thomson Computer Press.

Gordon (1986). "Before You Leap" User's Guide Gordon Group.

Hall, E. M. (1998). Managing Risk. Methods for Software Development, Addison-Wesley, Reading, MA.

Hamaker, J. W. (1994). "But What Will It Cost? The History of NASA Cost Estimating", In: "Readings in Program Control," edited by Francis T. Hoban, William M. Lawbaugh and Edward J. Hoffman, NASA SP-6103, Washington, DC (available online at http://www.jsc.nasa.gov/bu2/hamaker.html)

Jones, C. (1986). Programming productivity MCGraw-Hill

Jones, G. W. (1990). Software Engineering, J. Wiley, N. York.

Kaufmann, A., M. M. Gupta (1985). Introduction to fuzzy arithmetic: theory and applications, Van Nostrand Reinhol, NY.

Kitchenham, B. (1990). Software Development Cost Models, in P. Rook, ed., Software Reliability Handbook, Elsevier, NY

Laplante, P. (2000). Dictionary of Computer Science, Engineering and Technology, CRC Press.

Madachy, R., Selby, R (1999). The COCOMO 2.0 Software Cost Estimation Model.

Mares, M. (1994). Computation over fuzzy quantities, CRC Press, Boca Raton, FL.

Matson, J. E., B. E. Barrett, J. M. Mellichamp. (1994). Software Development Cost Estimation Using Function Points, IEEE Trans. on Software Engineering, 20, 4, 275-287

Miller, I.R., J.E. Freund, R. Johnson (1990). Probability and Statistics For Engineers, Prentice-Hall

Mood, A. M., Franklin, A G. and Duane, C. B (1963). Introduction to the Theory of Statistics, McGraw-Hill, New York,

Musilek, P., W. Pedrycz, G. Succi, M. Reformat, N. Sun. Interactive Simulation of a Fuzzy Model for Software Cost Estimation. Proc. of the 2001 International Conference on Simulation and Multimedia in Engineering Education, Phoenix, AZ, Jan 2001

Musilek, P., W. Pedrycz, G. Succi, and N. Sun, On the Sensitivity of COCOMO II Software Cost Estimation Model, In Proc. 8th IEEE Symposium on Software Metrics, Ottawa, 2002, pp. 13-20

Pedrycz, W., F. Gomide (1998). An Introduction to Fuzzy Sets. Analysis and Design, MIT Press, Cambridge, MA.

Pillai, K. and Sukumaran Nair, V.S. (1997). A Model for Software Development Effort and Cost Estimation. IEEE Transactions on Software Engineering 23(8) 485-497.

Putnam, L. H. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem, IEEE Transactions on Software Engineering 4 (7), 345-361.

Putnam, L. H. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem, IEEE Trans. on Software Engineering, 4, 4, 345-361

Rubin, H. A. (1983). "Interactive macro-estimation of software life cycle parameters via personal computer: a technique for improving customer/developer communication:" proc.

Symp. On application & assessment of automated tools for software development, IEEE, San Francisco.

Rubin, H. A. (1984). "Macro and micro-estimation of maintenance effort: the estimacs maintenance models" IEEE.

Rubin, H. A. (1985). "A comparison of cost estimation tools" proc. 8th Int. Conf. Soft. Eng. IEEE.

Stutzke, R. D. (1996). Software Estimating Technology: A Survey, Science Applications International Corp.

Zadehn, L. A. (1975). The Concept of a Linguistic Variable and Its Application to Approximate Reasoning Information Sciences, 8, 199—251.

Appendix A:

COCOMOII data

Project ID	TDEV	PM	Size	PREC	FLEX	RESL	TEAM	PMAT	RELY	DATA	RUSE	DOCU	CPLX
1	48	2040	134.47	4.96	5.07	7.07	4.38	7.80	0.92	1.28	1.07	1	0.73
2	36	240	57.72	2.48	3.04	6.36	2.19	7.80	0.82	1.28	0.95	1	0.73
3	18	423	35.70	4.96	5.07	4.24	3.29	5.46	1.1	0.9	0.95	1	1.34
4	24	218	30.36	3.72	5.07	2.83	2.19	5.46	1.26	0.9	1.07	1	1.34
5	10	79	24.00	4.96	4.05	4.24	3.29	4.68	1.1	0.9	1	1	1.34
6	16	724	110.50	4.96	5.07	4.24	2.19	4.68	1.1	1	1.15	1	1
7	16	539	75.50	4.96	5.07	4.24	2.19	4.68	1.1	1	1.15	1	1
8	12	88	11.19	3.72	5.07	5.65	4.38	4.68	1.1	0.9	1.07	1	1.17
9	24	1063	52.00	3.72	5.07	2.83	2.19	6.24	1.26	1.14	1.07	1	1
10	12	230	26.18	6.2	4.05	5.65	4.38	4.68	1.1	1.14	1	1	1
11	14	87	30.00	1.24	1.01	2.83	0	7.02	0.92	1.07	1	1	1.085
12	16	126	60.00	1.24	1.01	2.83	0	6.24	0.92	1.07	1	1	1.085
13	12	176	28.56	6.2	4.05	4.24	3.29	6.24	1.1	1	1	1	1
14	12	130	25.00	6.2	4.05	5.65	3.29	5.46	1.26	0.9	0.95	1	1.34
15	14	38	9.10	6.2	3.04	2.83	0	4.68	0.92	0.9	1	1	1.34

Project ID	TIME	STOR	PVOL	ACAP	PCAP	AEXP	PEXP	LTEX	PCON	TOOL	SITE	SCED	TURN	EST PM
1	1	1.05	1.15	1.19	1.15	1	1.14	1	1	1.29	0.93	1	1.07	982.57
2	1	1	0.87	1.19	1.34	0.81	0.91	0.91	1	1.17	0.93	1	1	208.15
3	1.29	1.17	1.15	0.85	0.88	0.88	1.14	1.145	1	1.17	1	1	1	218.36
4	1.11	1.46	1	0.85	0.88	0.81	0.91	1	1	1.17	0.93	1	1.07	206.05
5	1.11	1.05	1.15	0.71	0.76	0.88	0.91	1	0.9	1.17	0.93	1	1	138.03
6	1.245	1.05	1	0.85	0.88	0.81	0.91	1	0.9	1.17	0.93	1.43	1	727.55
7	1.0825	1.05	1	0.85	0.88	0.81	0.91	1	1.12	1.17	0.93	1.43	1	474.64
8	1.375	1.17	1	1	1	1.1	1	1	0.9	1.29	0.93	1.14	0.87	57.85
9	1.46	1.46	1.15	0.85	0.88	0.81	1.14	1.145	1	1.17	1	1	1.07	363.60
10	1.055	1	1	1	1	0.88	1	1	1	1.29	0.93	1.43	0.87	225.39
11	1.055	1.17	0.87	1	1	0.88	0.91	0.91	0.9	1.17	0.86	1	1.07	79.38
12	1	1.05	0.87	1	0.88	0.88	0.91	0.91	1	1.17	0.86	1	1.07	130.67
13	1.29	1.17	1	0.85	0.88	1	1.14	1	1	1.17	0.93	1	0.87	186.18
14	1.63	1.17	1	0.71	0.76	0.81	0.91	0.91	1	1.17	1	1	1	141.81
15	1.11	1.17	1.15	0.78	0.76	0.81	1.19	1.2	1	1.45	0.86	1	1	45.01

Appendix B

USArmy

No.	S(KLOC)	T(months)	E(PM)
1	85	35	1176
2	450	34	5112
3	341	40	4992
4	163	18	1908
5	200	24	2124
6	80	24	288
7	200	24	2064
8	126.5	17	1248
9	250	38	576
10	75	12	336
11	85	12	240
12	49.8	30	852
13	324	24	1680
14	173.5	33	444
15	100	34	588

Yourdon

No.	S(KOLC)	T(months)	E(PM)
1	40	35.2	81.8
2	70	24.1	142
3	132.2	32	116.4
4	23	16.4	21.6
5	34.5	16.4	130.2
6	14	15.1	29.7
7	25	16.9	61.6
8	52.1	16	21.9
9	7.1	11.3	20.5
10	25.9	16	13.4
11	16.3	14.1	14.2
12	17.4	11.6	11.9
13	33.9	12.3	39.9
14	55.8	27	16.9
15	21	12.4	23.7
16	7.58	9.97	6.1
17	8.78	19	28.6

Belady-Lehman

No.	S(KLOC)	T(months)	E(PM)
1	30	12	77
2	11.164	8	51
3	17.052	9	46
4	140	31	462
5	47.377	13	241
6	229	36	1665
7	401.099	24	1022
8	712.362	28	2176
9	58.54	28	723
10	80.99	12	527
11	94	42	673
12	18.775	32	199
13	14.39	17	227
14	35.057	19	71
15	11.122	8	43
16	6.092	8	47
17	5.342	4	14
18	12	8	60
19	19	10	50
20	25.271	12	169
21	20	14	106
22	12	9	57
23	7	9	195
24	13.545	17	112
25	14.779	7	67
26	30	68	1107
27	69.2	35	852
28	486.834	67	11758
29	220.999	61	2440
30	128.33	10	673
31	32.026	36	136
32	15.363	7	37
33	4.747	3	10

COCOMO63

Adjusted KLOC	Person Month
1.9	9
1.98	5.9
2.14	7.3
2.5	8
3	73
3.7	40
3.9	61
4	43
4.4	20
5.3	6
5.3	14
6.2	8
6.3	18
6.7	57
6.9	8
8.2	41
9.1	38
9.4	88
10	122
10	15
12	55
13	98
13	82
15	12
15	237
16	33
18	321
19.5	45
20	218
21	70
22	1075
22	230
23	36
24	79
24	176
25	130
27	958
28	83
28	50
30	423
30	87
32	106

34	47
37	201
38	523
40	605
46	240
48	387
50	1063
57	126
75	539
90	453
91	156
109	724
113	2040
132	243
249	160
252	2455
261	702
287	6400
311	1272
320	11400
966	6600